A Distributed Cache Architecture for Quality-of-Service Routing in Communication Networks

A thesis submitted in fulfillment
of the requirements for the degree of
Doctor of Philosophy

By
Mohammad Rezvan

Department of Computer Science
University of Canterbury
Christchurch, New Zealand
December 2000

Abstract

The rapid advances in telecommunication and Internet technologies is the driving force behind the emergence of a new generation of network-oriented multimedia applications such as Internet telephony and real time video applications. To successfully deploy these new applications, the communication networks should be QoS capable so that network resources such as bandwidth can be reserved during the lifetime of an application. One of the most fundamental elements of a QoS-capable network is QoS routing. The computing load caused by frequent execution of such algorithms, especially in large networks, is a concern. Additionally, QoS routing algorithms need up-to-date network topology and state information to compute the QoS routes. This mandates the regular distribution of the state and topology information across the network. The overhead traffic caused by frequent update and redistribution of state and topology information, especially in large networks is a concern.

In this thesis, we propose a distributed cache architecture to reduce the route computing load caused by on-demand execution of the QoS routing algorithms, assuming a bandwidth-based QoS model.

The distributed cache architecture has been designed to easily scale to large networks. In addition, we show that such an architecture helps to increase the robustness of QoS routing in the presence of inaccurate network state information caused by long network state update intervals. This means that the proposed distributed cache architecture can also reduce the overhead traffic caused by the frequent distribution of the network state information, while achieving a good performance.

To further utilize the advantages offered by the distributed cache architecture, we propose several novel techniques to improve its performance.

Firstly, we introduce a distributed technique called *cache snooping*. The goal of cache snooping is to alleviate the effects of the changes in the network states so that the accuracy of the cached routes is increased. In this way, the distributed cache architecture can reduce the route computing load more effectively. Cache snooping selects and monitors those segments that are more likely to be reused by arriving calls. Monitoring is performed by sending snooping packets across the network links.

The second proposed technique is called *route freezing*. While cache snooping has been designed to increase the accuracy of the cached routes in a statistical fashion, route freezing has been designed to provide 100% accurate routes that are not affected by changes in the network states. By manipulating the normal tear down procedure that is performed when a call ends, route freezing creates frozen cached routes.

The third proposed technique is called *route borrowing*. Normally, a cached route can be reused only at its terminal points, where the route starts and ends. Route borrowing allows the long end-to-end cached routes to be partially reused from their intermediate points as well as the end points. The goal of route borrowing is to increase the likelihood of re-using cached routes by arriving calls. This means that fewer calls have to be routed by on-demand computing so that the route computing load is even reduced even further.

We propose and evaluate realistic and practical solutions that can be deployed in real life large networks. We use stochastic discrete-event simulation to evaluate the performance of the proposed distributed cache architecture and its associated techniques. We consider realistic network topologies, routing algorithms, traffic models, and topology aggregation techniques.

Contents

Glossary

ATM	Asynchronous Transfer Mode
BGP	Border gateway Protocol
BW	Bottleneck Bandwidth
EGP	Exterior Gateway Protocol
FIFO	First-In-First-Out
IP	Internet Protocol
LFU	Least Frequently Used
LRC	Least Recently Computed
LSDB	Links State DataBase
MFU	Most Frequently Used
MRC	Most Recently Computed
MRU	Most Recently Used
OSPF	Open Shortest Path First
PNNI	Private Network-to-Network Interface
QoS	Quality of Service
RIP	Routing Information Protocol
RSVP	Resource Reservation Protocol

List of Figures

List of Tables

I have been privileged to have the best mother in the world

To Mamina

Chapter 1

Introduction

1.1 Quality-of-Service (QoS) in Communication Networks

The advent of broadband networking technology has dramatically increased the capacity of communication networks up to thousands of megabits per second. Added capacity makes it possible to support such applications as video conferencing, scientific visualization, Internet telephony, multimedia mailing, medical imaging, and interactive distance learning. These emerging applications, however, require the network to support guaranteed Quality-of-Service (QoS). Different applications require different QoS. Some applications require stringent end-to-end delay, while some require a minimal transmission rate. Although numerous service models have been proposed to deal with QoS requirements, most of them can be described as a function of delay, bandwidth, and/or throughput.

To satisfy the required QoS for individual applications, networks should reserve resources such as bandwidth during the lifetime of each application [Bra, Diff-url, Laz91]. This is in contrast to the majority of currently supported services by widely used communication networks such as Internet. Although in recent years there are emerging multimedia applications such as the Internet telephony, still best effort services with no QoS support have dominated the Internet.

To successfully deploy new services to meet an application's QoS requirements, communication networks should employ resource management mechanisms at

application, transport and network levels. One of the fundamental elements of such resource management architecture is routing.

1.2 Routing in Communication Networks

Routing is a fundamental part of an hierarchical resource management framework. In its simplest form, the goal of routing is to find a route from a source to a destination in accordance with the traffic's service requirements and the network's resource constraints, while maximizing network performance and minimizing the cost of the route based on a link cost function (e.g., number of hops). Finding a route between a source and a destination is referred to as unicast routing. However, in practice sometimes a set of routes (i.e., a tree) needed to be found between a source and a group of destinations. This is referred to as multicast routing [Dee90, Kom93]. One may consider unicast routing to be a subset of multicast routing. Before focusing on QoS routing, we outline several major design issues and trade-offs involved in every routing architecture.

1.2.1 Static vs. Dynamic Routing

Static routing is inherited from telephone networks. In static routing, the routing decision is made without considering the network state. Consequently, while trying to find a route between nodes, a static routing scheme does not take into account the dynamic conditions of the network state and uses a static link cost function to find the route. While easy to implement, these schemes are subject to serious limitations and perform poorly when the network load varies.

Dynamic routing schemes use real time network state information to compute a route [Ash81, Mat96]. Therefore, they can react appropriately to changes in the network state. This makes them suitable for QoS routing. Dynamic route selection algorithms make use of real time network state information to maintain a real time big picture of the topology and the state of the network. A dynamic routing scheme involves two basic tasks: firstly, collecting the network state information and keeping it up to date; secondly, using the network state information to compute a route. Different dynamic routing schemes may require different state information to be advertised (e.g., hop count, bandwidth, or buffer space).

1.2.2 Link-State vs. Distance-Vector Routing

As mentioned in Section 1.2.2, dynamic routing schemes use network state information to compute routes. The network state information can be maintained and used in two ways: global and local. From this perspective, there are two categories of routing schemes [Tan96]: firstly, the link-state routing schemes that maintain and use the network state information in a global fashion; secondly, the distance-vector routing schemes that maintain and use the network state information in a local fashion.

The link-state routing schemes flood the routing information to all nodes in the network [Moy95, Gup95, and Gar95]. However, each node sends only that partition of the routing table that describes the state of its own links. Therefore, each node has a global view of the entire network. In distance-vector routing schemes, each node sends its entire routing table, but only to its immediate neighbours. In other words, link-state routing schemes send small updates to the entire network [Cai2000], while distance-vector routing schemes send large updates only to immediate neighbours [Mal95].

Link-state routing schemes are less prone to routing loops than distance-vector routing schemes, because they create a consistent view of the network [Ber92, Gup95]. On the downside, link-state routing schemes can cause significant control traffic in the network. They are also computationally expensive and need more CPU and memory resources than distance-vector routing schemes.

1.2.3 Unicast vs. Multicast Routing

The unicast routing problem can be defined as follows. Given a source node s and a destination node d, and possibly an optimisation goal (e.g., balancing the network load or minimising the number of hops), find the best route from s to d. The multicast routing problem can be defined as follows. Given a source node s and a destination set D, and possibly an optimisation goal, find the best tree covering s and all members of the set D [Dee90]. In many cases, unicast and multicast routing problems are closely related [Lor99]. In fact, in many cases multicast routing can be viewed as a generalisation of unicast routing. In this thesis, we focus on unicast routing. While multicast routing is beyond the scope of this thesis, the solutions and techniques developed in this thesis may be applicable to support multicast routing.

1.2.4 Routing Strategies

Based on the way network state information is maintained, and the way a routing architecture uses the network state information for computing routes, three routing strategies can be found: source routing, distributed routing and hierarchical routing. In the following sections, we detail these strategies.

1.2.4.1 Source Routing

Under the source routing strategy, the routing decision is made only at the source node [Gup95] where the call has arrived. Each node maintains the complete global state, including the network topology and state information for the entire network.

Source routing offers several advantages. Firstly, it achieves its simplicity by a centralized approach to route computation. This centralized approach guarantees loop free routes. Secondly, many source routing algorithms are conceptually simple and easy to implement. Thirdly, it enhances the flexibility of the network since different route computing algorithms can co-exist in different source nodes in the network without interfering with each other. This facilitates the deployment of multi-vendor network architectures because there is no danger of instability due to routing loops.

There are also several disadvantages associated with source routing. Firstly, the global state maintained at every node has to be updated frequently enough to cope with the dynamics of network states and to provide the routing algorithms with sufficiently accurate data. This can cause excessive traffic overhead, especially in large networks. Secondly, the propagation delay of the network links, coupled with a lower update frequency for the network states (to avoid excessive overhead) causes inaccuracy in the network state information maintained at the source nodes. This can greatly reduce the performance of the routing. Thirdly, the computing overhead at the source nodes is a concern. This is especially true in the case of multicast routing or when multiple QoS requirements are required (assuming a QoS capable network). In general, source routing has a scalability problem, but as we discuss later, this can be addressed by topology aggregation and combining the source routing with the hierarchical routing strategy. The ATM PNNI standard is based on source routing [Gup95, ATM96].

1.2.4.2 Distributed Routing

Distributed routing is sometimes referred to as hop-by-hop routing. Under the distributed routing strategy, the route computation load is distributed among all the intermediate nodes between the source and the destination. All nodes across a route contribute to find a segment of the route [Nel90, Gar95]. Hence, the routing response time can be made shorter and greater scalability is achieved [Sal97, Ach2000]. Most existing distributed routing algorithms require each node to maintain a global network state, based on which routing decision is made on a hop-by-hop basis. Some flooding-based algorithms do not require any global state to be maintained. The routing decision is made based on the local states.

Distributed routing algorithms, which depend on the global state, share more or less the same problems as source routing algorithms. It is also very difficult to design efficient distributed heuristics for the NP-complete routing problems, especially in the case of multicast routing, because there is no detailed topology information available. In addition, when the global states are inconsistent across the network nodes, loops may occur. A loop can be detected when a node receives the routing message for the second time, an generally makes the routing fail. The Internet protocol (IP) uses a distributed routing scheme [Cra97].

1.2.4.3 Hierarchical Routing

Hierarchical routing has been used to cope with the scalability problem of source routing in large networks [Tsu88, Tsa89, Hao98]. The ATMM PNNI standard has adopted such a routing model [Atm96, Ahn2000]. Hierarchical routing scales well because each node only maintains a partial global state, where groups of nodes are aggregated into logical nodes. The source routing algorithms are used at each hierarchical level. Hence, hierarchical routing retains many advantages of the source routing. It also incorporates some advantages of distributed routing because the route computing occurs in a shared fashion among multiple nodes.

However, because the network state and topology information are aggregated, additional inaccuracy is introduced, which has a significant negative impact on the performance of the routing [Gue97-1]. We discuss this issue in more detail when we focus on QoS routing.

In general, hierarchical schemes are rather inflexible with regard to routing in the presence of changing network loads. Hence, it is generally agreed that limiting to just two levels of hierarchy is sufficient to provide significant scalability in large networks

[Hao2000]. Incorporating two levels of hierarchy leads to a model in which a large network is partitioned into network domains [Kle77, Ala95, Cal97, Beh98]. The two levels of hierarchy are defined as follows.

1. Intra-domain level. At this level, the internal topology and network state of the domains are represented. Each domain is connected to the rest of the network via several border nodes. Each node in a domain has complete and detailed information about the topology and state information of the domain. The network nodes outside a domain have only an aggregated view of the domain topology and state information, achieved by applying topology aggregation discussed in Section 1.3

2. Inter-domain level. This level presents the connection between the domains. Only an aggregated picture of the network domains is represented at this level.

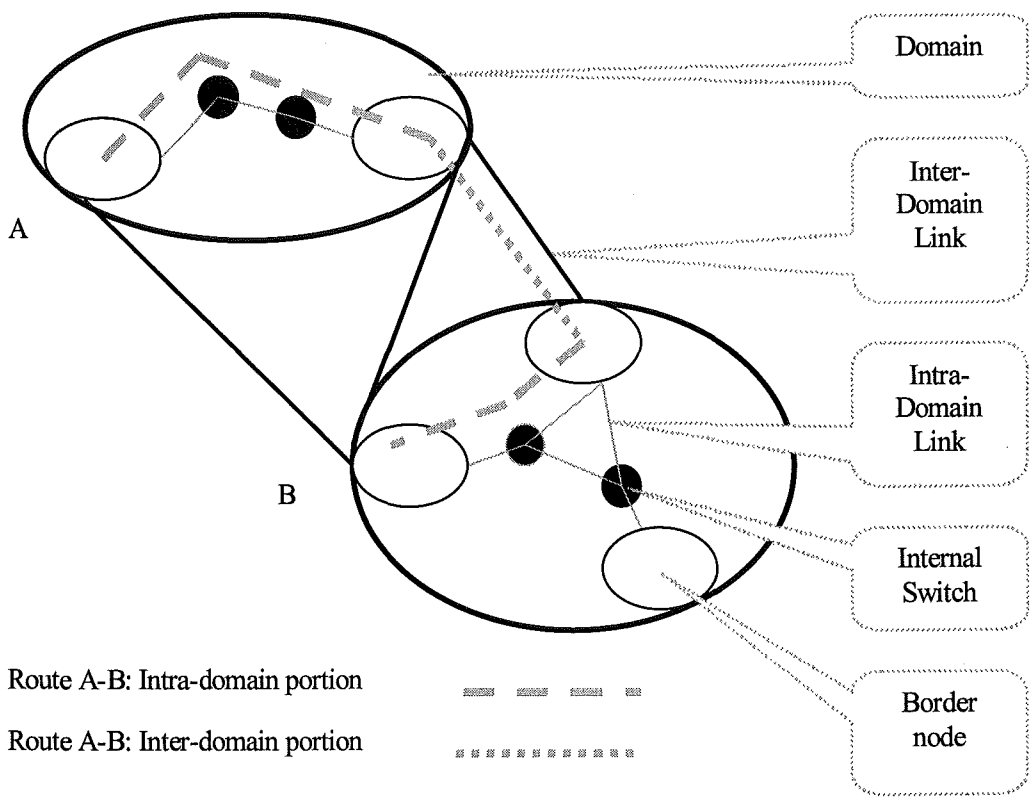


Figure 1.1: Inter-domain and Intra-domain routing

Assuming a two level hierarchy is in place, there are two types of routing protocols: inter-domain and intra-domain. Inter-domain routing protocols deal with the routing inside a domain, while intra-domain protocols deal with the routing between domains. IP-based networks such as the Internet use RIP, IGRP and more recently, OSPF for intra-domain routing [Hed88, Moy91]. BGP [Lou91, Rek95] and EGP [Bas97] have been used for inter-domain routing. The ATM PNNI standard has adopted a unified end-to-end approach in which a single protocol is used for both inter-domain and intra-domain routing [Gup95, Atm96]. Figure 1.1 shows a very simplified view of the inter-domain and intra-domain routing concept. As it shows, in a network with multiple domains, routes are potentially made up of several inter-domain and intra-domain segments. As we discuss later, in this thesis, we have adopted a two level network model coupled with source routing.

1.3 Topology aggregation

Topology and network state aggregation is one of the key techniques for achieving scalability in large networks (including QoS capable networks), since it can reduce the amount of network state information and the routing table size by orders of magnitude [Lee95-1, Gou98]. Not surprisingly, it is even proposed for routing in non-QoS networks [Ock77, Tsu88].

The topology aggregation is strongly associated with the hierarchical routing network model [Iwa98]. Throughout this thesis we have considered a network model with two levels of hierarchy so that a large network is partitioned into multiple domains. Therefore, we discuss the topology aggregation assuming only two levels of hierarchy. Topology aggregation is achieved by grouping the neighbouring network in a *compact* fashion. Consequently, outside nodes have only a brief image of the internal state of the aggregated domain. Thus, a routing domain can be considered by the outside world to be a single logical node with several ports connected to the outside. The term *port-to-port distance* that has been used in the literature, reflects the fact that network nodes outside of a domain make routing decisions based on the routing costs between different ports.

Topology aggregation introduces a two-way trade-off between the scalability on one side, and the accuracy of the network topology and state information on the other side. Scalability is provided in two ways:

1. Topology aggregation reduces the storage requirements for the network topology and state information. This is because when it is used in a network, each network

domain presents only a brief and simplified image of its internal topology and state information to other domains. This is very important, especially in very large networks with many domains.

2. It reduces the amount of overhead traffic caused by the regular distribution of the network state and topology information. This is an important issue especially when the network state information has to be re-distributed frequently.

However, scalability is achieved at the price of a loss of accuracy in the network state information. This lack of accuracy is an important issue because it can reduce the overall routing performance in two ways:

1. Computing the wrong route. This can happen because QoS route computing relies on the network state information, which is less accurate when aggregation is used in the network. Therefore, a computed route can fail to be set up successfully in the network.

2. Missing an available route. This can happen for a similar reason. In this case, the routing algorithm fails to find a feasible route because of inaccurate network state information.

There is a large body of research about topology aggregation in the literature. For example, in [Awe98-1, Awe98-4], Awerbuch studies the role of aggregation in extremely large networks. In [Awe98-2], Awerbuch gives presents theoretical foundations of applying aggregation to directed graphs. Also in [Awe98-3], Awerbuch explains a simulation test-based designed to evaluate the performance of different aggregation techniques. In [Lee-95-1, Wha95-2], Lee and Whay give a deep insight into the role of topology aggregation in ATM networks. Several aggregation techniques such as full-mesh, symmetric-star [Hao2000] and spanning tree [Wha95-1] have attracted more attention and have been under more extensive study. While topology aggregation is not the focus of this thesis, we have incorporated it in our network and routing model, as it is an essential part of large real-life networks. In this thesis we have used the full-mesh and star aggregation techniques, as detailed in Chapter 3.

1.4 Quality-of-Service (QoS) Routing

A core component of many recent proposals for QoS-capable networks is the ability to identify a route between a source and destination pair that can satisfy a set of QoS requirements for a particular application [Lee91, Mat95]. For traffic requiring stringent QoS guarantees, QoS routing computes routes that meet the resource constraints imposed by the user's QoS requirements [Cam94, Lee95-2]. From this perspective, QoS routing can be considered as a specialized and more complicated subset of routing. In general, as Figure 1.2 shows, a QoS routing architecture has four main functional components: signaling, route computation, and state information distribution/update.

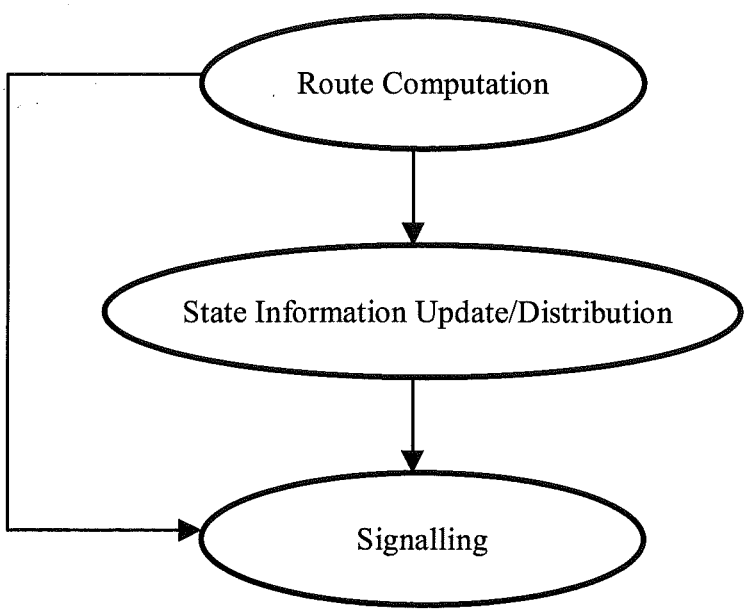


Figure 1.2: Three functional components of a QoS routing architecture

1 *Signalling*. This is required to establish and tear down routes, reserve resources in the network nodes and links across an end-to-end route, and distribute network state information across the network. For example, RSVP has been proposed as a signaling and resource reservation protocol for QoS capable Internet [Zha93].

2 *Route computation.* This is the core part of a QoS routing architecture. Assuming a unicast routing model, the duty of the route computation part is to compute a route between a pair of nodes that can satisfy one or more QoS requirements, such as bandwidth and/or delay. There may be a need for computing a QoS tree, if the network supports multicast routing. Ideally routes are computed by executing a routing algorithm against the network topology and state information. Naturally, if the network has adopted a source-based routing scheme, routes are computed in a centralized fashion based on a global image of the network. In contrast, if a hop-by-hop routing scheme has been adopted, routes are computed in a distributed fashion. In recent years, a variety of different routing algorithms have been proposed. In Section 1.5, we present an overview of several recently proposed QoS routing algorithms.

3 *State information distribution/update.* As mentioned, a QoS routing scheme should be dynamic. In other words, it should take the dynamics of the network state into account while computing QoS routes. To realize this, network nodes should maintain an up-to-date image of the network state topology. On the other hand, variations in factors such as call arrival rate, call holding time, requested QoS by individual calls, and distribution of the load in the network, causes the nodes' view of the network state to become obsolete. To alleviate this problem, all nodes in the network send periodic updates to the rest of the network. The network state update introduces a variety of challenges and design trade-offs, as discussed in Section 1.5. Figure 1.2 shows three main functional components of a QoS routing architecture. As can be seen, both route computation and state distribution components rely on the signaling component.

1.5 Important Aspects of QoS-Capable Networks

In this section, we overview the important aspects of a QoS-capable network. It is important to mention that we emphasize large-scale networks because scalability is likely to be a serious barrier to the deployment of QoS-enabled services in large networks such as the Internet.

Here, we do not intend to present a deep analysis, but rather to give the reader a wide perspective on major design issues. In subsequent chapters of this thesis, where we discuss and analyze our proposal, whenever necessary, any relevant design issues will be discussed in depth. It is worth mentioning that when we talk about QoS routing, by *route,* we mean an end-to-end connection between two user nodes; by *link,*

we mean a connection between two adjacent nodes (i.e., switches) inside the network. Therefore a *route* is made up of several *links*.

1.5.1 QoS Parameters

From an end-to-end perspective, there are two types of QoS parameters: additive and non-additive [Nag93, Cra97]. The end-to-end value of an additive QoS parameter is the sum of the partial values associated with individual links of that route. As an example, delay is an additive QoS parameter [Sal97]. The delay for an end-to-end route is sum of the partial delays associated with links across that route. On the other hand, an example of an end-to-end value of a non-additive QoS parameter is the largest or smallest value among all partial values associated with links across that route. For example, bandwidth is a non-additive QoS parameter.

From a user perspective, a call can request for one or more QoS parameters. It has been shown that finding a route that simultaneously satisfies multiple QoS parameters is NP-complete [Gar79]. Although studies such as [Che98-1, Kap2000] have presented several heuristics to find routes with multiple QoS constraints, they can be used only under very specific assumptions.

When considering a QoS model, care should be taken to avoid complexity. For example, the QoS extension to OSPF [Gue97-2] has adopted a simple, yet realistic bandwidth-based QoS model. As we will discuss in Chapter 2, we have adopted a bandwidth-based QoS model.

1.5.2 Execution Strategies for QoS Routing Algorithms

Regardless of their complexity and detailed characteristics, QoS routing algorithms can be executed in two different ways: on-line and off-line.

On-line route computing is also referred to as real-time or on-demand computing [Gaw95]. In on-line computing, the routing algorithm is executed upon the arrival of a new call in the network [Wan95-2, Lac98, Erg2000]. The main advantages of on-line computing are simplicity and the reduction in storage requirements. The disadvantage of on-line computing is that it can cause considerable processing overhead. This can cause scalability problems in large-scale networks, especially when the network is experiencing a high call arrival rate [Vog96, Por97].

Off-line route computing is also referred to as pre-computing. In off-line computing, route computing is essentially de-coupled from call arrivals. Upon a call arrival, a pre-computed route that can satisfy the requested QoS will be selected. There have been

several proposals on route pre-computing [Le95, Sha98, Ord2000]. For off-line route computing to be considered as a practical solution, several important issues should be carefully addressed. Firstly, each network node needs realistic criteria to determine the destinations to which the routes are pre-computed. In recent proposals all possible routes to all destinations have to be computed and stored in a QoS routing table. This may prove to be inefficient in terms of processing power and storage requirements. Secondly, changes in the network states can cause the pre-computed routes to become obsolete. There should be mechanisms in place to keep the pre-computed routes fresh. Such mechanisms can cause significant computing overhead, especially if there are too many pre-computed routes. Thirdly, in large networks with an hierarchical architecture, it may not be feasible to easily store pre-computed routes. This is because the network domains are subject to topology aggregation.

Route caching has been proposed to alleviate the on-line route computing load. Several studies [Pey97-1, Apo98-1, Apo98-2, Su2000] propose simple caching schemes. In Route caching, routes are computed in an on-demand fashion, then stored in a cache for potential use by further calls. Re-using the cached routes helps to reduce the computing load caused by on-demand computation of QoS routes. Issues such as cache content management, cache replacement policy, and cache update, are keys to achieving a good performance. As detailed in Chapter 2, a novel distributed cache architecture forms the foundation of our proposed QoS routing architecture.

1.5.3 An Overview of QoS Routing Algorithms

The survey in [Che98-2] presents a detailed description of different QoS routing algorithms. In this section, we briefly overview these algorithms. In accordance with our classification of the routing strategies presented in Section 1.2.4, we overview the proposals under three main categories: source routing algorithms, distributed routing algorithms, and hierarchical routing algorithms. For clarity, we refer to the different algorithms using the name of the first author of the corresponding published paper.

1.5.3.1 Source QoS Routing Algorithms

The major proposals for source-based QoS routing algorithms are as follows.

- The Wang algorithm [Wan95-1]. This algorithm finds a bandwidth-delay-constrained route by applying Dijkstra's shortest path algorithm. First, all links with a bandwidth less than the requested amount are eliminated so that any route in

the resulting graph will satisfy the bandwidth requirement. Then, the shortest route in terms of delay or hop counts is found. Also, for traffic with bandwidth requirements, Wang proposes the shortest-widest QoS routing algorithm. This algorithm has been adopted in this thesis. It first selects all routes with the maximum available bandwidth, then selects the one with the smallest hop count. This algorithm emphasizes minimizing resource consumption by selecting the route with the minimum hop count.

- The Ma algorithm [Ma97-1]. It has been shown that when a class of WFQ-like (Weighted Fair Queuing) scheduling algorithms has been used, the end-to-end delay, jitter, and buffer space bounds are not independent [Dem89, Zha90, Gol94]. Therefore, the problem of finding a route satisfying bandwidth, delay, jitter, and buffer space, which is normally NP-complete, can be simplified. It can be solved by a modified version of the Bellman-Ford algorithm in a polynomial amount of time.

- The Guerin algorithm [Gue97-1]. This study focuses on the bandwidth-constraint and the delay-constrained routing problems with inaccurate network state information. The model of the inaccuracy is based on the probability distribution functions. The authors propose heuristic algorithms to transform the global constraints into local constraints. This algorithm works with inaccurate network state information and is also suitable for use with hierarchical routing. One of the heuristic algorithms was extended by the authors to be used in networks with aggregation. A further study of QoS routing with inaccurate network state information based on the probability model was conducted in [Lor98]. Also in [Gue97-2] Guerin proposes a widest-shortest QoS routing algorithm, which has been adopted in this thesis. This algorithm first selects all feasible routes with minimum hop counts, then selects the one with the maximum available bandwidth. It emphasizes balancing the network load by selecting the least loaded (widest) route. However, it should be noted that selecting the route with maximum (or minimum) available bandwidth does not guarantee an optimal solution for routing. The arrival sequence of demands with different capacity requirements has an influence on whether the demands can be accepted.

- The Awerbuch algorithm [Awe93]. This is a throughput-competitive routing algorithm for bandwidth-based routes. The algorithm tries to maximize the average throughput of the network over time. It combines the functions of admission control and routing. Every link is associated with a cost function that is exponential to the bandwidth utilization. A survey for the competitive routing algorithms can

be found in [Plo95]. In this thesis, we have also adopted a QoS routing algorithm from this family [Hao2000].

1.5.3.2 Distributed QoS Routing Algorithms

A number of istributed QoS routing algorithms have been formulated:

- The Salama algorithm [Sal97]. A distributed heuristic for the NP-complete delay-constrained least-cost routing has been proposed. A cost vector and a delay vector are maintained at every node. The cost (delay) vector contains for every destination the next node on the least-cost (least-delay) route. A control message is sent from the source toward the destination to construct a delay-constrained route. Loops may be created as the control message chooses the least-delay routes. A loop is detected if the control message visits a node twice. Whenever this happens, the routing procedure returns to the node where the loop started, then resumes from there by changing the next hop along the least-delay route. It was proven that such a mechanism removes all loops, provided that the delay and cost vectors at all nodes are up-to-date. However, this assumption may be unrealistic in a network with a dynamic state.

- The Cidon algorithm [Cid97]. This work combines the process of routing and resource reservation together. Every node maintains the topology of the network and the cost of every link. When a node requires a route to be set up to a destination with certain QoS requirements, it finds a subgraph of the network that contains links that lead to the destination at a "reasonable" cost. The authors refer to such a subgraph as a *diroute*. A link is *eligible* if it has the required resources. Reservation messages are flooded along the eligible links in the diroute toward the destination and reserve resources along different routes. When the destination receives a reservation message, a route is established. The algorithm releases resources from segments of the route as soon as it learns that whether these segments are inferior to another segment.

- The Shin algorithm [Shi95]. This work proposes a distributed routing algorithm for delay-constrained routes. No global state needs to be maintained at any node. The algorithm floods routing messages from the source toward the destination. Each message accumulates the total delay of the route it has traversed so far. When a routing message is received by an intermediate node, the message is forwarded

only when one of the following conditions is satisfied: Firstly, if it is the first such message received by the node. Secondly, if it carries a better accumulated delay than the previously received messages. Once a message reaches the destination, it finds a delay-constrained route, which is the one it has traversed.

- The Chen algorithm [Che98-3]. This work proposes a distributed routing scheme based on *selective probing*. After a call arrives, probes are flooded selectively along those routes which satisfy the QoS. Every node only maintains its local state, based on which routing decisions are made. Each probe that arrives at the destination detects a feasible route. Based on the selected optimization criterion, one of the feasible route may be selected. This work also explores and derives algorithms for a variety of different QoS parameters including bandwidth, delay, jitter, and their combinations.

1.5.3.3 Hierarchical QoS Routing Algorithms

The only major proposal in this category is ATM PNNI protocol [Atm96]. In contrast to the previous proposals that are solely theoretical, the ATM PNNI is a well-established industrial standard. It is a comprehensive end-to-end QoS routing architecture, which allows users the freedom to incorporate their own routing algorithms. It combines hierarchical routing with source routing, assuming the presence of topology aggregation at all levels of the hierarchy. Each source node maintains a global view of the entire network. This means that each source node has fully detailed information about its local domain, while maintaining only an aggregated view about the rest of the network. The topology and the network state information is stored in a special database called a Link State DataBase (LSDB). The network state information is frequently redistributed across the network. Upon arrival of a call at a source node, based on the requested QoS parameter and the destination, the source node uses its LSDB to compute a primary skeleton route. The primary skeleton route is computed based on the detailed topology and state information about the local domain to which the source node belongs, and the aggregated topology and state information of other network domains. When the route setup message enters a network domain, the source (border) node of the domain where the route setup message has entered computes and fills that part of the route skeleton which belongs to its domains. Due to the inaccuracy of the network state and topology information at the original source node, it may happen that the route setup procedure fails at an intermediate domain. To cope with this problem, ATM PNNI employs a technique

called *crankback*. This causes the route setup procedure to roll back to the latest source (border) node from which the route computing has been initiated. Upon receiving the crankback message, the source node computes a new route without considering the original route skeleton for its local domain.

This process continues until the set up message reaches the destination. In this way, ATM PNNI actually combines source routing and distributed routing. As we see, several source nodes collaborate in computing an end-to-end route. This interesting approach alleviates the inaccuracy of the network state information caused by topology aggregation.

1.5.4 Accuracy of the Network State Information

Because QoS routing algorithms rely on the network state and topology information to compute QoS routes, the accuracy of this information is of paramount importance and can significantly influence the performance of QoS routing schemes [Gue97-1, Ord99]. Two factors affect the accuracy of the network state information: topology aggregation and network state update.

As detailed in Section 1.3, topology aggregation reduces the size of the network state and topology information so that the storage requirement and the overhead traffic caused by the distribution of the network state is reduced. In this way, it provides greater scalability [Guo98]. On the other hand, topology aggregation reduces the accuracy of the network state and topology information.

In a QoS capable network, regular re-distribution of the network state is required to provide an up-to-date view of the network for QoS routing algorithms. To accomplish this, each node periodically sends and receives network state information updates to and from other nodes respectively. This process can cause significant overhead traffic and scalability problems [Pey97-2, Hao2000, Kun2000]. Three key factors affect this overhead:

1. *Update size.* As discussed in Section 1.3, topology aggregation addresses this issue. Although topology aggregation reduces the storage requirements, it can only partially reduce the update overhead traffic. To be able to efficiently reduce the update overhead traffic, we need to control the update frequency as well as the update policy.

2. *Update frequency.* A higher update frequency causes more overhead traffic, but also increases the accuracy of the network state information, and a lower update frequency causes less overhead traffic, but also reduces the accuracy of the

network state information. As mentioned, the accuracy of the network state information is very important because it affects the routing performance [Gue97-1].

3. *Update policy.* In the network nodes, updates are triggered upon certain conditions, referred to as update policies. In general, update policies can be classified into two categories: change-based and timer-based policies. In a change-based update policy, an update is triggered by a significant change in link states. In timer-based update policies, a timer is used to either trigger an update at fixed intervals or to enforce a minimum interval between two consecutive updates.

The network state update introduces a trade-off between the overhead traffic and the accuracy of the topology and the network state information.

1.6 Trade-offs and Obstacles in Designing QoS Routing Architectures

To design an efficient and realistic QoS routing architecture several inter-related design issues and technical obstacles should be addressed. The performance of the QoS routing architecture is significantly influenced by the tailored trade-off between these design issues [Kur93]. We classify these issues and obstacles as follows.

1. *Scalability.* The scalability of a QoS routing architecture is a multi-dimensional issue with several important aspects [Guo98, Hao99]. Firstly, a realistic QoS architecture should be easily deployable in large networks. Here, it should be taken into account that a large network is potentially divided into multiple domains (i.e., two levels of hierarchy), where domains are potentially subject to topology aggregation. Secondly, route computing load is a concern, especially in large networks with a heavy load. This can create scalability problems by causing excessive computing load on the network switches. Thirdly, the overhead traffic caused by the distribution of the network state information, coupled with signalling requirements of a QoS routing architecture can cause scalability problems as the network size grows. We pay special attention to the issue in Chapter 2 of this thesis.

2. *Route computing load.* Traditionally, in a QoS routing architecture, upon arrival of a new call, a QoS route is computed. This on-demand route computing approach is understood to be a non-trivial barrier in designing large scale QoS capable networks, mainly because of the generated computing overhead [Apo98-1, Apo98-

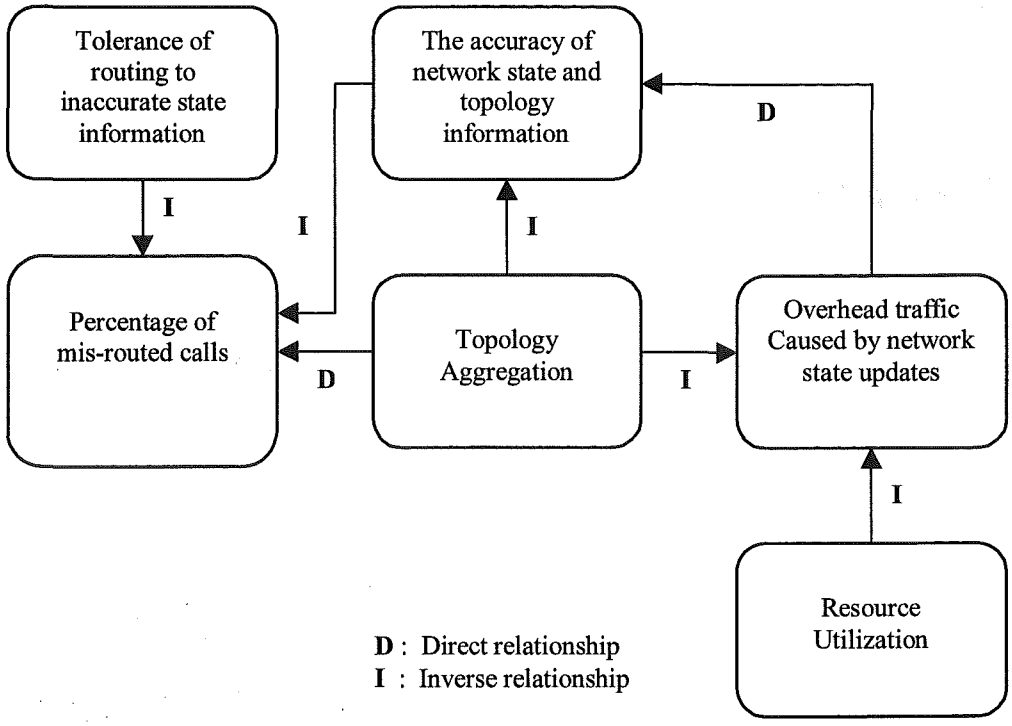


Figure 1.3: Interaction between different design issues in QoS routing

2]. This can be an even more complicated barrier when complex routing algorithms are used in the network. This can cause computing hot spots in large networks with high arrival rates. As mentioned in Section 1.5.2, route pre-computing and route caching have been proposed as potential solutions to reduce the route computing load. This issue is the main focus of this thesis.

3. *Resource utilization and load balancing.* This is related to route optimization. A poor resource utilization policy in a QoS routing architecture can increase the network state fluctuation (e.g., by selecting heavily loaded links) which causes more updates and therefore greater overhead traffic in the network. On the other hand, a pure emphasis on balancing the network load can cause bandwidth fragmentation, which can increase the call rejection ratio for future calls. The challenge is to provide a dynamic balance between resource utilization and balancing the network load. However, this issue is beyond the scope of this thesis.

4. *Overhead traffic.* As discussed, the adopted topology aggregation, the state update frequency, and the adopted update policy together control the amount of the state update traffic in the network [Noy90]. In addition, a QoS routing architecture

causes some overhead traffic due to its signalling requirements for operations such as route setup and tear down procedures. However, this signaling traffic is likely to be trivial in comparison to the overhead traffic caused by the distribution of the network state information. Here, we face a trade-off between the scalability in terms of overhead traffic and the accuracy of the network state information. Although we have not concentrated on this issue, our proposed solutions help to reduce the amount of traffic caused by the distribution of the network state information.

5. *Tolerance to inaccuracy.* Decreasing the traffic caused by the frequent distribution of the network topology and state information, coupled with topology aggregation, are two key techniques for achieving scalability in terms of storage requirements and overhead traffic in the network. The price of this scalability is loss of accuracy in the network state information stored in the network nodes [Lor98]. This can potentially reduce the overall routing performance [Gue97-1, Su2000], and appears in the form of wrong routing decisions. This includes calls that are routed, but are unable to satisfy the requested QoS, and calls that are not routed while feasible routes actually exist. Maintaining highly accurate network state information, especially in large networks, is neither realistic nor feasible. It is not realistic because it is a costly design in terms of the overhead traffic caused by frequent network state updates. It is not always feasible because the inaccuracy caused by the topology aggregation can not be compensated by any other technique. In addition, the propagation delay of network links can contribute to some extent in the inaccuracy of the network state information. Therefore the alternative and more realistic solution is to design a QoS routing architecture that can offer at least a reasonable performance in the presence of highly inaccurate network state information. In this thesis, we propose techniques to help increase the tolerance of QoS routing in the presence of inaccurate network state information.

Figure 1.3 shows the interaction between different design issues in a QoS routing architecture. Note that in Figure 1.3, mis-routed calls include calls that are routed but unable to satisfy the requested QoS and calls that are not routed while feasible routes actually existed.

1.7 The Thesis

This thesis proposes an end-to-end architecture to reduce the route computing load caused by on-demand computation of QoS routes. We put the basis of our proposal on an end-to-end QoS routing architecture that can be easily deployed across real and potentially large networks. Issues, such as the hierarchical structure of large networks, topology aggregation, the diversity of traffic in large networks, and inter-domain and intra-domain topologies are taken into account. For the sake of tractability, we consider a connection-oriented network model. We have adopted a bandwidth-based QoS model.

As discussed in Section 1.5, in QoS-capable networks, routes are computed upon arrival of calls. The main advantage of this on-demand approach is its simplicity. However, in large networks with high arrival rates, this approach can cause a significant computing load.

The pre-computing technique has been proposed and shown to be an effective solution to reduce route computing load [Le95, Sha98, Ord2000]. The principle is to compute routes as a background process and use them when a call arrives, therefore reducing the computing load upon each arrival. This thesis focuses on route caching to reduce the route computing load by re-using already computed routes. In route caching, a newly computed route is stored in a cache for possible use by future calls.

To the best of our knowledge, only two main studies have proposed route caching [Pay97-1, Apo98-1]. These two approaches are similar, with only minor differences in the adopted route selection policies and the route replacement policies. Upon arrival of a call at a node, the cache in that node is searched for a route that can satisfy the requested QoS parameter for the requested destination. If no such route is found in the cache, then a new route has to be computed. Because the cache size is limited, cache replacement policies should be used when the cache is full. In addition, when several feasible routes are found in the cache, efficient route selection policies are required to maximize network resource efficiency. While caching is a promising approach to reduce route computing load, we believe that recent proposals have taken very simplistic and primitive approaches, and several fundamental issues have received no attention. In addition, we identify several design problems in their approach. We classify these issues as follows:

- The architecture of large networks has not been taken into account. A realistic approach should consider that a QoS architecture needs to scale to large networks. As discussed throughout this chapter, large networks have an hierarchical architecture. Assuming a two-level hierarchy, a large network is divided into

multiple domains. The proposed caching architectures completely ignore this issue by assuming very small networks with a few switches.

- A centralized cache architecture has been considered by these proposals. Therefore, each network node maintains an independent cache, where it saves and reuses the routes. This approach causes several problems. Firstly, the cached routes can be reused only in one direction. For example, assuming two network nodes as *A* and *B*, a cached route between *A* and *B* that is stored in the cache at *A* can be reused only by calls that arrive at *A*. If a call arrives at *B* and requests a route to *A*, *B* has to compute and store a separate route to *A*. When the size of the network grows, this can cause significant redundancy. Secondly, although it is claimed in [Apo98-1] that a small cache size is efficient, the results are obtained using very small networks. When the size of the network grows, a centralized approach is likely to face scalability problems in terms of storage requirements. This is because the cache at each network node has to maintain a large number of routes to many destinations.

- The deployment of the centralized cache architecture is not feasible in a large network, which is partitioned into multiple domains (i.e., hierarchical architecture). As mentioned, the state and topology information of network domains is subject to topology aggregation. Therefore, network nodes are unable to cache all details of an end-to-end route that crosses several domains. The important question is: how can such an end-to-end route be cached efficiently? The centralized cache architecture clearly fails to answer this crucial question.

- The proposed cache architectures are tightly coupled with the network state information maintained at the network nodes. For example, in [Apo98-1], a route is stored in the cache as a list of node structures. Each node structure is a pointer to the corresponding entry in the link state database maintained of the node (a link state source based routing has been considered). Under this approach, different cache operations, such as updating and re-using the cached routes, rely on the network state information maintained by network nodes for making decisions. As discussed throughout this chapter, the network state information maintained by the network nodes always suffers from inaccuracy. Therefore, the inaccuracy of the network state information can have a negative impact on different cache operations.

In this thesis, we propose a novel *distributed cache architecture* to reduce the on-demand route computing load, while addressing the above-mentioned issues. As we discuss in more detail in Chapter 2, we assume a connection-oriented ATM-like network model, where routes are computed at the source node.

The performance evaluation of the distributed cache architecture and its associated techniques is based on an extensive set of stochastic discrete-event simulation scenarios, through which we can assess different aspects of our proposed architecture. We use realistic network topologies, aggregation techniques, routing algorithms, and traffic conditions. This allows us to draw realistic conclusions about the performance of the proposed cache architecture under various conditions that may occur in the operation of real networks.

1.7.1 Contributions

In summary, this thesis makes the following contributions.

- The first contribution of this thesis is the design and evaluation of a *distributed cache architecture* to reduce the route computing load caused by on-demand computing of QoS routes. The distributed cache architecture has been designed to scale to large networks, and stores end-to-end QoS routes in the form of inter-connected segments. Furthermore, it utilizes a distributed cache content management mechanism called *cache flushing*. A set of parameters and policies control the functionality of the architecture. The results indicate that the distributed cache architecture is quite effective in reducing the route computing load.

- As the second contribution, this thesis introduces a distributed technique called *cache snooping*. The goal of cache snooping is to alleviate the effects of the changes in the network states so that the accuracy of the cached routes is increased. In this way, the distributed cache architecture can reduce the route computing load more effectively. Cache snooping selects and monitors those segments that are more likely to be reused by arriving calls. Monitoring is performed by sending snooping packets across the network links. In fact, we have completely de-coupled the distributed cache architecture from the inaccuracy of the network state information stored in the network nodes. In other words, cache snooping does not rely on the potentially inaccurate network state information stored in the network nodes. Several parameters control the performance of the cache snooping.

- The third contribution of this thesis is to show that by adding the cache snooping to the distributed cache architecture, *the tolerance of the routing to inaccurate network state information is increased.* This is reflected by the lower bandwidth blocking ratio achieved under snooping when the network state is highly inaccurate. This suggests that the distributed cache architecture helps to reduce the overhead traffic caused by frequent distribution of the network state information without a negative side effect on the overall performance of the routing. This is an important result, because traditionally the overhead traffic caused by frequent distribution of the network state information has been recognised as a scalability problem, especially in large networks. On the other hand, less frequent distribution of the network state information causes inaccurate routing information and can seriously lower the performance of the QoS routing.

- The fourth contribution of this thesis is the introduction of a technique called *route freezing*. While cache snooping has been designed to increase the accuracy of the cached routes in a statistical fashion, route freezing has been designed to provide 100% accurate routes that are not affected by changes in the network states. By manipulating the normal tear down procedure that is performed when a call ends, route freezing creates frozen cached routes. All network resources across the network links remain allocated for a frozen route. To achieve a good performance and to minimize the potentially negative side effects of the route freezing on the performance of the routing, several parameters control the functionality of route freezing. We show that if tuned carefully, the route freezing significantly reduces the route computing load, especially when the network state information is highly variable.

- The fifth and the last contribution of this thesis is the introduction of a technique called *route borrowing*. Normally, a cached route can be reused only at its terminal points, where the route starts and ends. Route borrowing allows the long end-to-end cached routes to be partially reused from their intermediate points as well as the end points. The goal of route borrowing is to increase the likelihood of re-using cached routes by arriving calls. This means that fewer calls have to be routed by on-demand computing so that the route computing load is even reduced even further. We show that route borrowing significantly increases the effectiveness of the distributed cache architecture in reducing the route computing load. This is especially true for large networks.

1.7.2 Thesis Outline

This thesis has been prepared in eight chapters. In Chapter 2, we introduce and detail the proposed distributed cache architecture and its associated techniques such as snooping and freezing. Chapter 3 details the simulation assumptions and describes our choice of network topologies, aggregation techniques, traffic characteristics, routing algorithms and performance measures. The performance of different aspects of the proposed cache architecture is evaluated in a structured fashion. In Chapter 4, we evaluate the basic functionality of the distributed cache architecture without considering techniques such as cache snooping, route freezing and route borrowing. Chapter 5 evaluates the performance and impact of the cache snooping. Chapters 6 and 7 evaluate the impact of route freezing and route borrowing respectively. Chapter 8 concludes the thesis and outlines potential directions for future research.

Chapter 2

A Distributed Cache Architecture for Quality-of-Service (QoS) Routing in Large Networks

2.1 Introduction

This thesis proposes and evaluates a distributed cache architecture to reduce the route computing overhead caused by on-demand calculation of QoS routes in QoS capable networks. We have adopted many concepts from various technologies and standards such as ATM and the Internet, but we have not tightly coupled our proposed distributed cache architecture with them. This enables our proposed techniques to be easily integrated into different technologies and industrial standards.

The remainder of this chapter is organized as follows. Sections 2.2.1 to 2.2.3 detail our adopted network, routing and QoS models, respectively. As our proposed distributed cache architecture interacts closely with the network and the routing models, it is crucial to present a clear view of these models. Section 2.2.4 details how elements of our distributed cache architecture fit into the network, and how routes are stored across the cache elements in a distributed fashion. The internal structure of the cache elements is outlined in Section 2.2.5. Sections 2.3 and 2.4 detail all necessary algorithms and policies for the route caching and route selection procedures, respectively. In Section 2.5, we propose a cache content management technique called cache flushing. Section 2.6 presents a novel technique called cache snooping, which has been developed to increase the accuracy of the cached routes. Sections 2.7 and 2.8 introduce the route freezing and route borrowing techniques which further improve

the performance and the efficiency of the distributed cache architecture. Section 2.9 briefly outlines the reliability and fault tolerance aspects of the architecture. Section 2.10 is conclusions.

2.2 Details of the Distributed Cache Architecture

The adopted network, routing and QoS models are presented, then the different aspects of the distributed cache architecture are detailed.

2.2.1 Network Model

The proposed distributed cache architecture is designed to scale to large networks. Therefore, we have considered a large network that is partitioned into several domains. Each domain is connected to the rest of the network through its border nodes. Domains are subject to topology aggregation, a commonly used technique to reduce the overhead traffic caused by network state updates. Therefore, nodes inside a domain have a detailed view of the topology and network state information of the domain, while the rest of the network has only an aggregated view of the domain. Under such a network model, two different topologies co-exist:

1. Intra-domain topology, which represents the internal topology of network domains. This topology is subject to topology aggregation.

2. Inter-domain topology, which defines how network domains are connected to each other. This topology presents a big picture of the whole network.

2.2.2 Routing Model

We have assumed a source-based link state routing model similar to the ATM PNNI standard [ATM96]. As mentioned in chapter 1, several features make source routing attractive for using in QoS capable networks. Firstly, it achieves its simplicity by a centralized approach to route computation. This centralized approach guarantees loop free-routes. Secondly, many source routing algorithms are conceptually simple and easy to implement. Thirdly, it enhances the flexibility of the network since different route computing algorithms can co-exist in different source nodes in the network without interfering with each other. This facilitates the deployment of multi-vendor network architectures because there is no danger of instability due to routing

loops. Calls arrive at border nodes of domains. Routes are computed and set up between two border nodes. All routes are bi-directional.

In link state routing, network state information is periodically distributed in the network, so that each border node maintains a link state database (LSDB) containing topology and network state information. Because network domains are subject to aggregation, each border node maintains detailed topology and network state information about its local domain, while storing only brief information about other domains in the network. The switches in the network periodically update the network state and topology information on their corresponding links. As discussed in Chapter 1, border nodes use these updates to maintain an up-to-date view of the network in their LSDB.

In a large network with multiple domains, a route potentially crosses several domains. Therefore, initially only a skeleton route is computed. This is because the border nodes in a domain have only an aggregated view of the topology and the network state information of remote domains. When a route is computed, it should be set up in the network. As the route setup procedure proceeds, when the route setup request enters a domain at a border node, the border node constructs the detailed structure of the route for that domain.

2.2.3 QoS Model

In this thesis, we assume a bandwidth-based QoS model, so that calls request for bandwidth as their QoS parameter. This simple, yet realistic, model can represent a broad range of applications [Ma97-2], which enables us to focus on other aspects of the distributed cache architecture. A route can satisfy the requested bandwidth of a call if its bottleneck bandwidth is at least as large as the requested bandwidth. We refer to such a route as *feasible*. By bottleneck bandwidth, we mean the smallest available bandwidth across all links of a route. For example, in Figure 2.1, the route A-B-C-D has a bottleneck bandwidth of 2, the route A-B-E-C-D has a bottleneck bandwidth of 5, and the route A-B-F-C-D has a bottleneck bandwidth of 1.

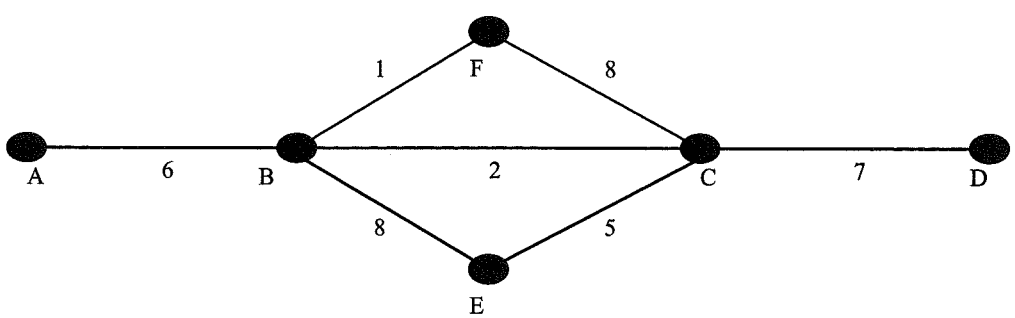


Figure 2.1: Bottleneck bandwidth on different routes

2.2.4 Deployment of the Distributed Cache Architecture in the Network

Figure 2.2 shows how our distributed cache architecture is deployed in a large network. Every border node in a network domain maintains a *cache element*. These cache elements are connected together by means of pointers (network addresses) and form the basis of our distributed cache architecture. Each cached route is stored across several cache elements in a *distributed* fashion and in the form of several *segments*. By segment, we mean the part of a route that is laid between two adjacent border nodes. Therefore, routes enter the border nodes in the form of an *ingress* segment and leave the border nodes in the form of an *egress* segment. When a route crosses a border node, the cache element at the border node stores information about both the *ingress* and *egress* segments. For example, in Figure 2.2, assume that upon arrival of a call at the border node A, a route has been successfully computed between the border nodes A and C. This route also passes through the border node B and is stored across cache elements at the border nodes A, B, and C as follows. The cache element at the border node A stores the A-B segment, the cache element at the border node B stores the A-B and the B-C segments, and the cache at the border node C stores the B-C segment. Note that the caches at two ends of a route store only one segment.

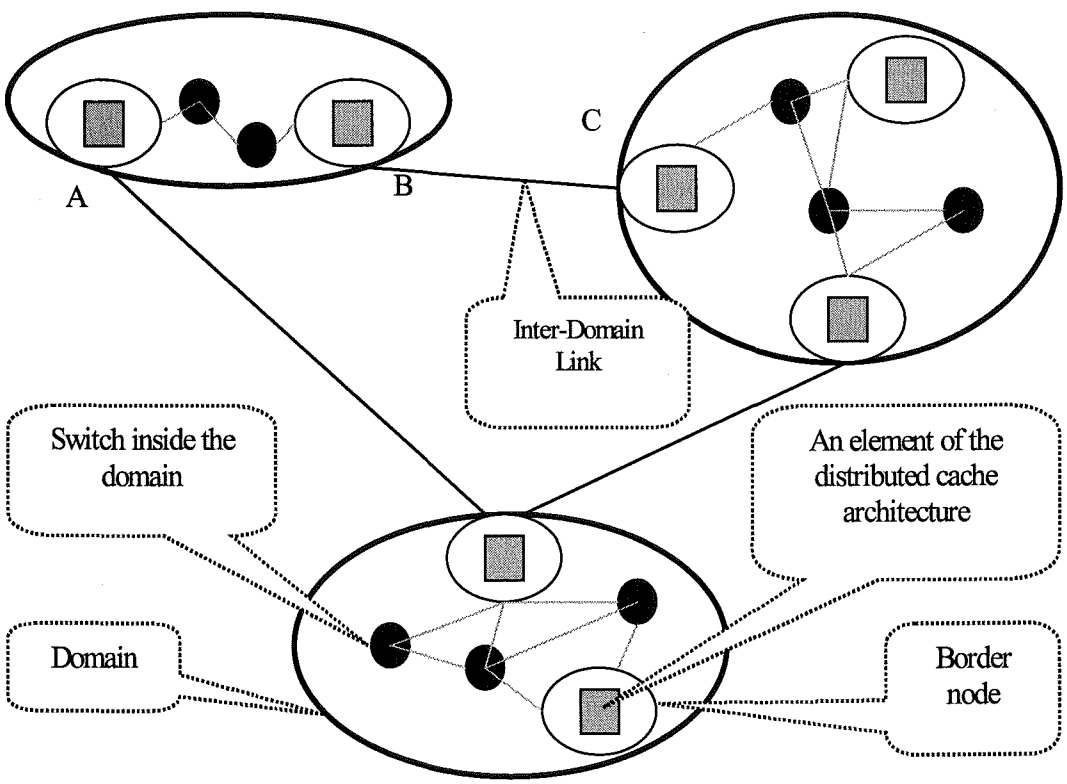


Figure 2.2: Deployment of the distributed cache architecture in a network with multiple domains

2.2.5 Internal Structure of Cache Elements

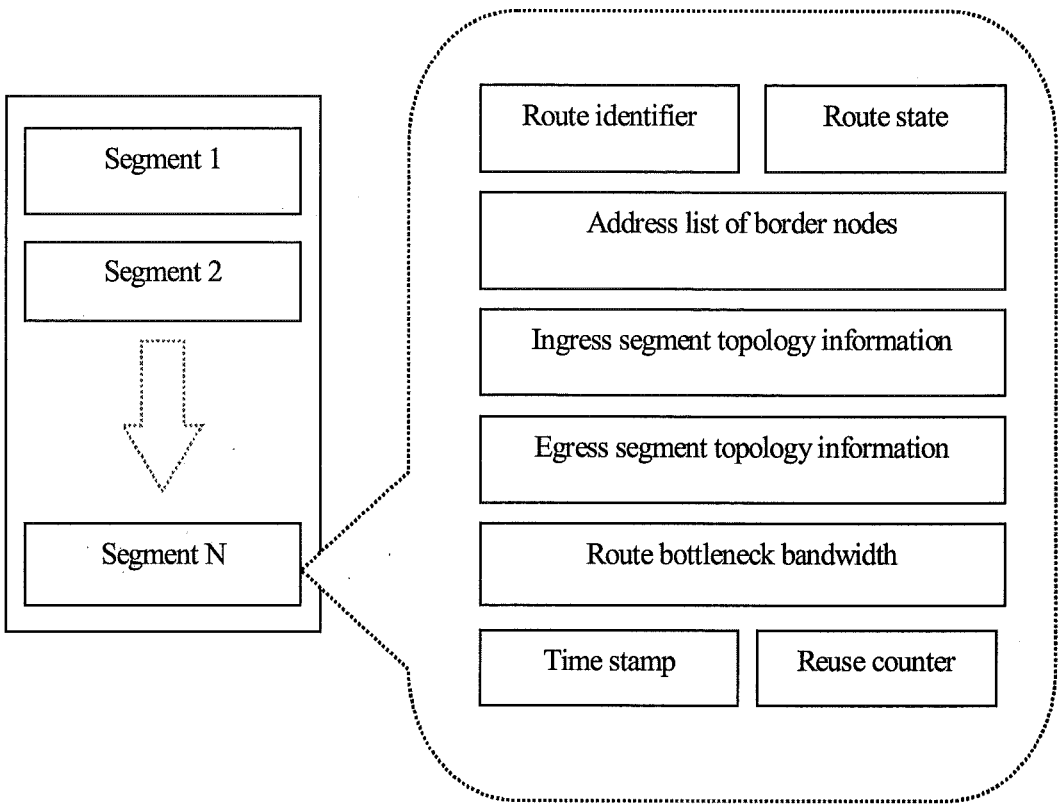


Figure 2.3 shows the internal structure of a cache element at a border node.

As Figure 2.3 shows, each entry in a cache element is made up of several different fields as follows.

- *Route identifier.* This is a number assigned to a route when it is cached. The combination of this number and the address of the source node help to identify different segments of a route that are stored across different cache elements.

- *Route state.* A cached route can be in several states. It can be *in use* by a call, be *available* for arriving calls, or be *stale.* As explained later, different cache operations and techniques set this field appropriately for the correct operation of the cache architecture.

- *Address list of border nodes.* This list is used to traverse a route. It contains the addresses of the source, destination, and all intermediate border nodes that a route has crossed. This list is arranged as a bi-directional linked list so that it can be accessed and traversed in a bi-directional fashion.

- *Route bottleneck bandwidth.* The QoS parameter used in our architecture.

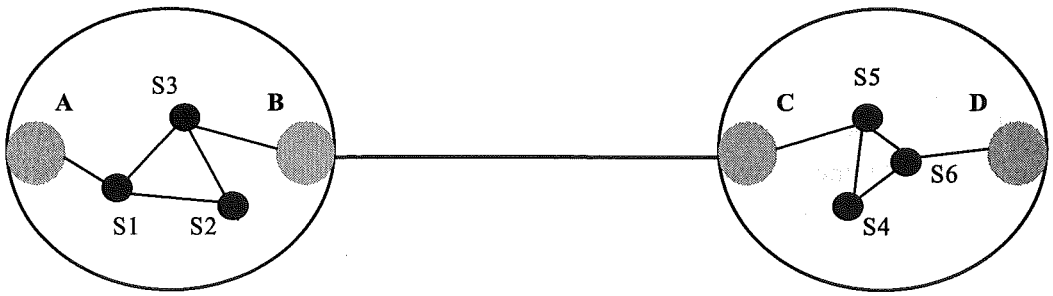
- *Topology information of ingress and egress segments.* For each route, each cache element stores the topology information of up to two segments connected to it. In fact, the cache element in the first and the last border nodes store only one segment, while caches in all the intermediate border nodes store two segments. The topology information is used by operations such as route selection or cache snooping. Therefore, once a route is cached, it can be used without relying on topology information stored in the LSDB. In other words, the cache architecture is completely de-coupled from the LSDB.

- *Time stamp.* The field keeps the time when the route was computed and stored in the cache for the first time. It may be used by a variety of time-based policies and techniques.

- *Re-use counter.* This is a counter that is increased by one each time the cached route is re-used successfully.

2.3 Route Caching Procedure

Considering our routing and network model, the process of caching a new route proceeds in parallel with the route setup procedure. Upon arrival of a call at a border node, if there is no feasible route found in the cache element of the border node, a route has to be computed on-demand. In accordance with the source-based link state routing model, based on the topology and network state information stored in its LSDB, and the selected routing algorithm, the source node computes a primary route skeleton that crosses one or more domains. As the route setup message crosses the border nodes of the domains, upon successful setup, each segment is stored by the cache element at the corresponding border nodes at its two ends. As route caching proceeds, the route state is set to *"in-use"* in all corresponding cache elements. This is to avoid the route to be used by another call, while it is being setup in the network. This process continues until the last segment of the route is successfully cached and the route setup message reaches the destination border node. At this stage, the route



1. Call arrives at border node A.
It requests for b unit bandwidth to the border node D.

2. The cache element at border node A is searched. There is no cached route from A to D with b bottleneck bandwidth.

3. Based on its local LSDB, A computes the route A-s1-s3-B-C-D.
(A can see only a single pseudo link between C and D due to aggregation)

4. The A-s1-s3-B segment is setup in the network successfully.

5. The cache elements in A and B save the A-s1-s3-B segment.
Segments are labeled as "in-use".

6. The B-C segment is setup successfully.

7. The cache elements in B and C save the B-C segment.
Segments are labeled as "in-use".

8. C uses its local LSDB to fill the C-D skeleton as C-s5-s6-D.

9. The C-s5-s6-D is setup successfully in the network.

10. The cache elements in C and D save the C-s5-s6-D segment.
Segments are labeled as "in-use".

11. D sends back the route setup packet to A, indicating successful setup.
As this packet crosses through C, B, and A the address list of
border nodes Is saved in the correspond cache elements .

12. A receives the setup packet and call begins.

Figure 2.4: A detailed scenario for route caching procedure

setup message is sent back to the source indicating a successful call setup. As this message comes back across the network, the address list of border nodes will be stored across all corresponding segments in the cache elements and the route state is set to "*in-use*" in all cache elements across the route. If the route setup procedure fails at any stage, a failure message is sent back to the source node. As this message comes back to the source node, all the corresponding segments in the cache elements across the route will be labelled as "*stale*". Figure 2.4 details a simple scenario for the route caching procedure. Note that the caching process always assumes that cache elements at the border nodes have free space. This is because we have designed a novel cache content management/replacement technique called *cache flushing* that works as a background task and always maintains some free space in the cache elements at the

border nodes. We detail this technique in Section 2.5. As a general rule, in the distributed cache architecture, whenever a cache entry in a border node has to be accessed, regardless of the type of operation (e.g., storing a segment, snooping, or flushing), the corresponding cache element is *locked.* After the operation is finished, the cache element will be *unlocked.* This is to avoid multiple operations simultaneously accessing the cache element. In addition, there is a FIFO buffer associated with the cache element at every border node. This buffer is used to save the control messages sent by other cache elements during different operations.

2.4 Route Selection from the Cache

When a new call arrives at a border node, based on the requested bandwidth and the destination address, the cache element at the border node is searched for a feasible route. A route is feasible if its bottleneck bandwidth is equal to or larger than that requested by a call, it is in the *"available"* state, and it reaches the requested destination. If no feasible route is found in the cache, a route has to be computed. If only one feasible route is found in the cache, the route setup will be started as follows. Starting from the cache element at the source border node, where the route is found, the first segment of the route is extracted from the cache element and is set up in the network. If the segment is set up successfully, the corresponding entry in the cache element is labelled as *"in-use"* and route setup proceeds to the next border node. As setup proceeds across border nodes, the address list and the route identifier fields are used to keep track and find the route information and the next border node. This process continues until the last segment of the route is set up. If at any stage the route setup fails, all corresponding cache entries will be labeled as *"stale"* and an on-demand route computing will be started. When the call ends, the tear down procedure starts. At the end of the tear down procedure, when all allocated bandwidth for the route is released, the corresponding segments of the route in the cache elements across the network are labelled as *"available"*. Figure 2.5 details a simple scenario for the route selection procedure. For simplicity, in Figure 2.5 we have assumed that the selected route is not obsolete so that the route setup procedure does not fail.

2.4.1 Route Selection Policies

Upon arrival of a call at a border node, if more than one feasible route is found in the cache element of the border node, one of them should be selected to go through the setup process. In such a situation, we use one of the following route selection policies:

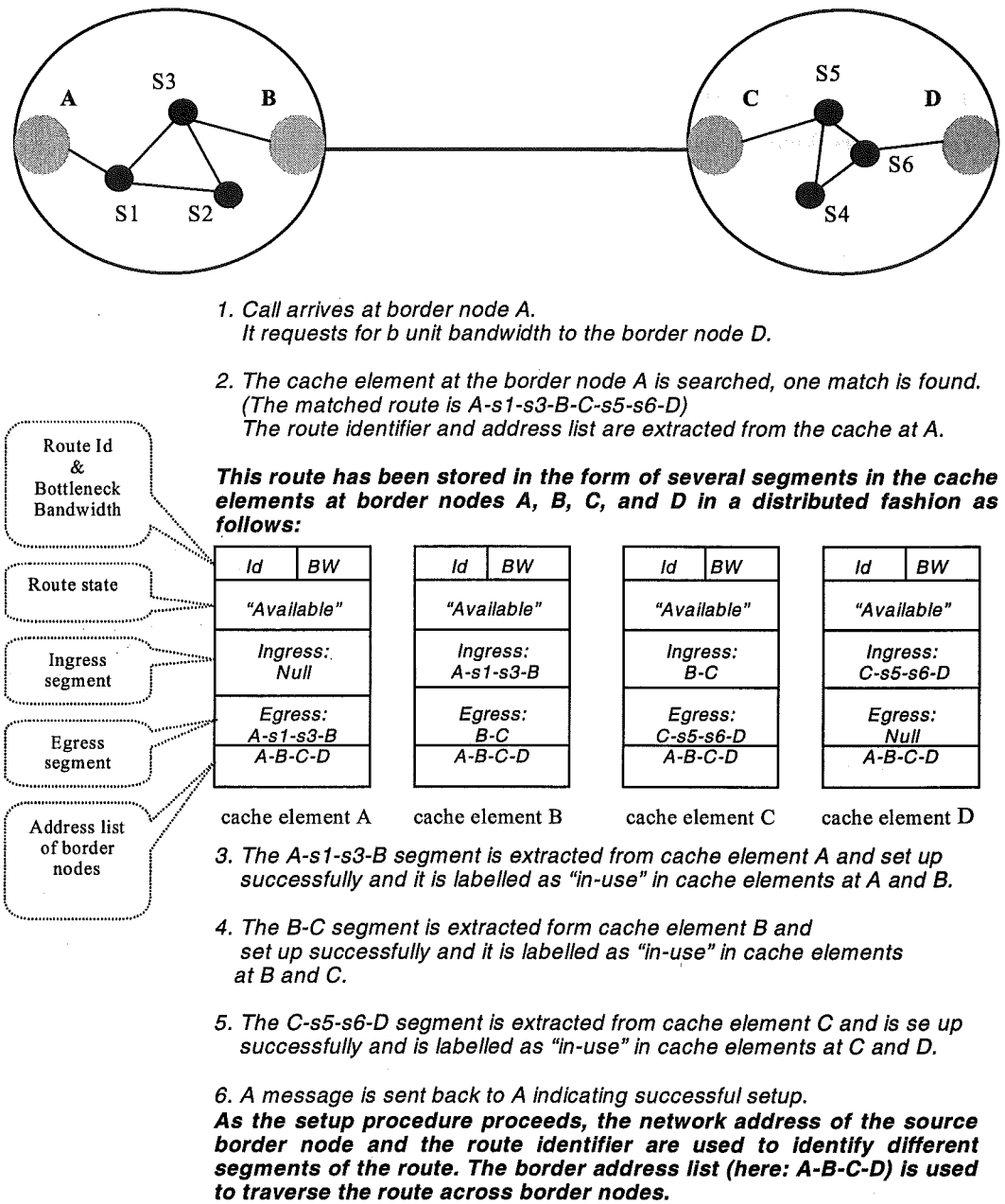


Figure 2.5: A detailed scenario for route selection from cache

- *Most Recently Computed (MRC).* This policy aims to minimize the probability of selecting an obsolete cached route by choosing the *youngest* route. The principle is that the most recently computed route has been subject to less fluctuation in the network states. This policy uses the time stamp of the cached routes.

- *Least Frequently Used (LFU).* This policy attempts to choose the *least popular* route, therefore increasing the availability of more popular routes for future calls.

To make decisions, this policy uses the reuse counter associated with each cached route.

- *Widest.* This policy chooses the widest of all feasible routes. Its goal is to balance the network load. It also attempts to reduce bandwidth fragmentation.

- *Tightest.* This policy attempts to choose the best fit. It leaves wider routes for future calls with possibly higher bandwidth requirements.

After a route is selected based on the adopted cache selection policy, it goes through the setup process.

2.5 Cache Flushing

As the size of cache elements is limited, a replacement mechanism is essential for the distributed cache architecture. The replacement mechanism is required when a cache element at a border node is full and a new segment needs to be stored in that cache element, so that the new segment replaces an already cached one. The straight on-demand approach is to perform the replacement when a new route needs to be cached and the cache is full [Apo98-1]. We believe that this on-demand approach is not very suitable for our distributed cache architecture, where there is a cache element at each border node. This is because different border nodes are subject to different traffic conditions. Consequently, a cache element at a particular border node may have free space, while at the same time, another cache element at another border node is full. Therefore, we have developed *cache flushing* that works for each border node as a background task. The principle of cache flushing is to flush one or more segments (and their corresponding routes) out, whenever the cache element at a border node becomes full. Therefore, flushing occurs at different cache elements independently[1]. Whenever the cache element at a border node becomes full, the flushing procedure starts and proceeds as follows:

1. The cache element is locked. As mentioned in Section 2.3, whenever a cache entry in a border node has to be accessed, regardless of the type of operation, the corresponding cache element is locked. After the operation is finished, the cache element will be unlocked. This is to avoid multiple operations simultaneously accessing the cache element.

[1] Flushing occurs independently for each cache element, but each time flushing is initiated in a cache element, other cache elements across the network will also be affected. This is because of the transmission of flushing messages between cache elements.

2. Based on a *flushing policy*, N_{flush} routes are selected to be flushed out (i.e., to be labelled as stale). N_{flush}, called the flushing size, is a flushing parameter that indicates the number of routes that should be flushed out during cache flushing.

3. For each flushed route, a message is sent to the cache elements of neighbouring border nodes that hold other segments of the route. Upon receiving the flushing message, the cache elements at neighbouring border nodes flush out the corresponding segments of the route. This process continues until all segments of the route are flushed out.

4. The cache element is unlocked to be available for other cache operations.

2.5.1 Flushing Policies

Flushing policy is an important factor and can significantly affect the performance of the distributed cache architecture. We have considered the following flushing policies:

- *Least Recently Computed (LRC).* This policy attempts to flush the oldest routes. An older route is more likely to be obsolete because it has been subject to more fluctuations in the network states. The route time stamp is used by this policy.

- *Least Frequently Used (LFU).* This policy attempts to flush those routes that are less likely to be used by future calls. It uses the re-use counter.

2.5.2 Flushing Size

Flushing size is another important parameter for cache flushing. A larger N_{flush} causes less frequent flushing, but more overhead traffic during each flushing. The overhead traffic is caused by the transmission of flushing messages between the cache elements at different border nodes. On the other hand, a smaller N_{flush} causes more frequent flushing, but less overhead traffic during each flushing. Adopting a very large N_{flush} causes a large number of cached routes to be flushed during each flushing. This may decrease the overall performance of the routing. However, adopting a very small N_{flush} may cause too frequent flushing and can cause instability in the cached routes. This, especially in large networks with higher propagation delays, may decrease the performance of the distributed cache architecture due to selecting the

flushed routes whose corresponding flushing messages are not yet received by all corresponding cache elements. As detailed in Chapter 4, we evaluate the effects of flushing size under a variety of different conditions.

2.6 Cache Snooping

Changes in the network states can cause cached routes to become obsolete (i.e., unable to offer their associated bottleneck bandwidth). Selecting obsolete routes from the cache adversely affects the routing performance. In addition, because the distributed cache architecture stores routes in the form of interconnected segments, an end-to-end route becomes obsolete when even one of its segments is obsolete. To address these problems, we propose a technique called *cache snooping*. The principle of cache snooping is to periodically *snoop (i.e., check up)* those cached segments that are more likely to be used, and keep them as up-to-date as possible. Snooping is not performed for segments that are being used by a call.

The concept of snooping has been used in computer architecture for providing cache coherency. In the field of computer architecture, cache memory is often used in conjunction with the main memory to reduce the memory access time. Because of the limited size, only a small part of the main memory can be mapped and saved in the cache memory. One of the technical challenges introduced by using the cache memory, especially in multiprocessor architectures, is keeping the cache contents the same as the content of the main memory. This is widely known as the *cache coherency problem*. Cache snooping has been introduced[2] as a solution to address this problem [Li89]. Under snooping, each cache memory continuously monitors the address bus of the main memory for its corresponding address range to detect the possible changes in that part of the main memory that is mapped in the cache. Upon detecting a change a variety of different actions can be taken. For example, the corresponding cache entries can be updated, or can be deleted.

The proposed distributed cache architecture faces a similar coherency problem. Here, the changes in the network states can cause the cached routes to become obsolete. We propose cache snooping to alleviate this problem by increasing the coherency (i.e., accuracy) of the cached routes according to the changes in the network states. Although the name of the technique may look similar to that adopted in the computer architecture research community, at the detailed level, our proposed

[2] We should emphasise that we do not intend to focus on detailed technical aspects of the cache coherency problem in the field of computer architecture. The goal is to make only a conceptual comparison between the proposed cache snooping technique for the distributed cache architecture and the snooping technique used in computer architecture.

solution is quite different from the snooping used in the field of computer architecture. We associate several parameters with cache snooping as follows:

- *Snooping Interval.* This defines how often snooping is initiated for the cache element at a border node.

- *Snooping Policy.* This defines the criterion for selecting segments for snooping each time snooping is initiated in the cache element at a border node.

- *Snooping Size.* This defines how many segments in a cache should be snooped each time snooping in initiated in the cache element at a border node.

To efficiently incorporate cache snooping in the distributed cache architecture, the snooping interval, snooping policies, and snooping size are parameters that should be carefully evaluated. We detail these parameters in subsequent sections. For now, assume I_{snoop} represents the snooping interval. Every I_{snoop} seconds, each cache element performs snooping only for its egress segments as follows:

1. The cache element is locked.

2. Based on a snooping policy, up to N_{snoop} egress segments are selected. N_{snoop} defines a limit on the number of segments that are selected for snooping. For each selected egress segment, the topology information is extracted from the cache element, and a *snooping packet* is sent across the segment in the network.

3. Upon receiving the snooping packet, each network switch enters the available bandwidth on its corresponding link in the packet and forwards the snooping packet to the next switch. This continues until the snooping packet reaches the end of the segment (i.e., another border node) and is sent back.

4. At the border node, where the snooping has been initiated, the smallest bandwidth value of the segment is extracted from the snooping packet. If the smallest bandwidth value is larger than or equal to the route bottleneck bandwidth, the snooping has been successful for this segment. Otherwise snooping fails because the route can not satisfy its associated bottleneck bandwidth. If snooping fails, the corresponding route is obsolete and should be deleted. This is done by sending messages to the cache elements that store other segments of the route.

5. The cache element is unlocked.

Figure 2.6 details a simple scenario for cache snooping.

2.6.1 Snooping Interval

The snooping interval (I_{snoop}) represents a trade-off between the snooping overhead and the accuracy of the cached routes. Cache snooping can cause computing and traffic overhead. The computing overhead is caused by operations such as accessing and searching the cache, and segment re-computing (in the case of active snooping). The overhead traffic is caused by the transmission of the snooping packets. A very large I_{snoop} causes less overhead but it can not increase the accuracy of the cached routes efficiently. On the other hand, a very short I_{snoop} may cause considerable snooping overhead but it may increase the accuracy of the cached routes more efficiently.

2.6.2 Snooping Policies

As mentioned, the goal of cache snooping is to increase the accuracy of the cached routes with minimum overhead. However, there is little benefit in snooping the cached routes that are not likely to be used by arriving calls. Therefore, a snooping policy is required to select only the cached routes that are more likely to be used. We have incorporated the following policies:

- *Most Frequently Used (MFU).* This policy assumes that a route that is used more frequently in the past is more likely to be used again. The route time stamp is used by this policy.

- *Most Recently Used (MRU).* This policy assumes that a route that is used more recently is more likely to be used again. This policy uses the reuse counter of the cached route.

- *Widest.* This policy assumes that a wider route can satisfy a broader range of requests. Therefore it is more likely to be used in the future.

2.6.3 Snooping Size

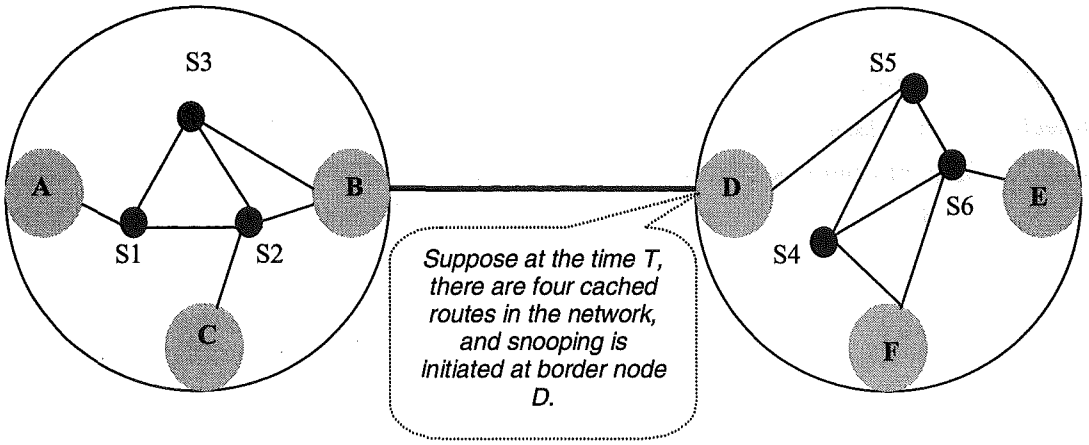
The snooping size (N_{snoop}) defines a limit on the number of segments that should be snooped in the cache element of a border node. Based on the adopted snooping policy, up to N_{snoop} segments are selected each time snooping is initiated in a cache element.

Therefore, if the total number of *"available"* segments in a cache element is more than or equal to N_{snoop}, then N_{snoop} segments will be selected. If the total number of *"available"* segments in a cache element is less than N_{snoop}, then all of them will be selected. A very large N_{snoop} can cause significant snooping overhead, especially when a cache element contains a large number of segments. However, a very small N_{snoop} is not likely to increase the accuracy of cached routes efficiently. This can lead to a poor overall routing performance.

2.6.4 Active Snooping

As discussed, cache snooping attempts to reduce the likelihood of selecting obsolete routes by deleting them from the cache. It causes little computing overhead, but may cause some overhead traffic due to the transmission of messages between cache elements of the border nodes. In this section, we propose an alternative approach to cache snooping called *active snooping.*

Active snooping attempts to *repair* the cached routes that are obsolete (i.e., partially damaged). This is accomplished by re-computing the obsolete segments. Therefore, it still can cause some computing overhead, especially when the network states are highly variable. However, in the long term, it may decrease the on-demand route computing load. Segment re-computing is similar to route computing, except that it is simpler and causes less computing overhead. In addition, segment re-computing does not suffer from the lack of accuracy caused by topology aggregation, because it relies on local information using network topology and state information stored in LSDB of the border node, where the snooping has started. Once a new segment is computed, it is stored in the corresponding cache element so that the end-to-end route is not deleted. If segment re-computing fails, the route is labelled as stale, by sending messages to the cache elements that hold other segments of the route. Active snooping attempts to increase the accuracy of the cached routes by repairing obsolete segments of routes.



Internal structure of a cache element at the border node D at time T, when snooping is initiated at the border node D:

ID	Bandwidth	State	Ingress segment	Egress segment	Address list
101	12	"In-use"	B-D	D-s5-s6-E	A-B-D-E
234	10	"Available"	B-D	D-s5-s6-F	A-B-D-F
132	11	"Available"	B-D	NULL	C-B-D
198	3	"Available"	B-D	D-s5-s4-s6-E	A-B-D-E

The corresponding end-to-end routes for the above route IDs are as follows:
101 : A-s1-s3-B-D-s5-s6-E
234 : A-s1-s2-B-D-s5-s6-F
132 : C-s2-s3-B-D
198 : A-s1-s2-s3-B-D-s5-s4-s6-E

Assume N_{snoop} = 1 and "widest" snooping policy is adopted.

1. *Cache element at the border node D is locked, then cache is searched for the widest route.*
2. *The route whose ID is 101 is widest but it is "in-use" state. The second widest route is 132, but it is terminated at D and has no egress segment. Therefore, route 234 is selected.*
3. *The egress segment topology information (D-s5-s6-F) is extracted from the cache element and assembled into a snooping packet. The packet then travels through relative links in the network.*
4. *As this packet is sent across the network, the available bandwidth on the relative links is recorded in the packet. When the snooping packet reaches F, it is sent back to the border node D. (Note that if ,multiple segments were selected, the snooping packets would be sent simultaneously)*
5. *The border node D, where snooping is initiated, extracts and defines the smallest recorded bandwidth in the snooping packet. If this number is less than the bottleneck bandwidth of the route, the route is obsolete, otherwise the route is healthy.*
6. *If the route is obsolete a message is sent to the border nodes E and B. Each border node, labels the corresponding segment of the route in its cache element as stale and passes the staling message to the next border node. This continues until all segments of the route are labelled as obsolete.*
7. *The cache element at the border node D is unlocked.*

Figure 2.6: A simple scenario describing cache snooping

2.7 Route Freezing

Under normal circumstances, when a call ends, the tear down procedure releases the allocated bandwidth for the call across the network links and the corresponding cached route is labelled as *"available"* for potential use by subsequent calls. With this approach, there is no guarantee that the cached route can satisfy its associated bottleneck bandwidth once the route is selected by a call. This is because changes in the network states can cause cached routes to become obsolete (i.e., unable to offer their associated bottleneck bandwidth). Selecting obsolete routes from the cache adversely affects the routing performance. In Section 2.6, we proposed cache snooping to improve the accuracy of the cached routes. However, this increases the accuracy of the cached routes only in a statistical fashion. In other words, cache snooping increases the probability of obtaining a valid route from the distributed cache architecture but it does not provide 100% accuracy for the cached routes.

In this section, we propose a technique called *route freezing*. The principle of route freezing is to manipulate the normal tear down procedure so that when a call ends, the allocated bandwidth across the network links for the call remains untouched and the corresponding route in the cache is labelled as *"frozen"*. The obvious advantage of such an approach is that once a frozen route is selected by a new arrival, it is 100% guaranteed that the cached route can satisfy its associated bottleneck bandwidth. In fact, a frozen route does not need to go through a setup procedure because the allocated bandwidth across the network links remains untouched. Therefore, route freezing can improve the overall performance of the distributed cache architecture, especially when network states fluctuate significantly, as a frozen route is completely isolated and shielded from changes in the network states.

One may find a conceptual similarity between route freezing and the Virtual Path (VP) in an ATM network. However, route freezing is rather different. Firstly, it has been proposed to increase the accuracy of the cached routes. This means that route freezing is coupled with the distributed cache architecture. Secondly, as we detail shortly, the functionality of route freezing is controlled by several parameters so that it can be incorporated in a flexible fashion.

Route freezing should be incorporated in the distributed cache architecture carefully and efficiently. To accomplish this, we have associated several parameters with it as follows.

- Freezing routes on a permanent basis may lead to the consumption of much of the available bandwidth on the network links. This can significantly decrease the overall network capacity. Therefore, we introduce a *freezing period* (T_{freeze}) that defines how long a route can be kept frozen.

- Attempting to freeze all cached routes in the network is certainly not efficient or practical. This is especially true for very wide routes with a very large bottleneck bandwidth because freezing such routes (especially over a long time) can significantly decrease the overall network capacity. We introduce a freezing parameter called the *freezing bandwidth threshold* (B_{freeze}). This allows us to freeze only a certain group of routes whose bottleneck bandwidth is under a certain threshold.

- In addition to freezing the bandwidth threshold, we need to put a limit on the number of routes that are selected for freezing. Therefore, we introduce the *freezing size* (N_{freeze}). The freezing size defines the maximum number of frozen routes in the cache element at a border node.

Route freezing is an end-to-end operation and each cache element at a border node initiates the route freezing only for routes that have been computed in that border node. Each border node maintains a small *freezing table* and a *freezing counter*. Each frozen route has an entry in this table with two fields containing the route ID and a *freezing timer* respectively. The freezing counter indicates how many routes at a given time, are frozen at a border node. Assuming route freezing has been incorporated into the distributed cache architecture, when a call ends, the following steps are performed.

- If the route ID is found in the freezing table, then its freezing timer is checked. If its freezing timer has expired (i.e., it is equal to or larger than I_{freeze}), a tear down procedure is initiated, the route is deleted from the freezing table and the freezing counter is decreased by one. As tear down proceeds across the network, segments of the route are labelled as *"available"*. If the freezing timer has not expired, all segments of the route will be labelled as *"frozen"*. This is accomplished by sending a message to the corresponding cache elements at the border nodes that the route has crossed.

- If the route ID is not found in the freezing table, then the freezing counter is checked. If the freezing counter is smaller than N_{freeze}, *AND* the route bottleneck bandwidth is smaller than B_{freeze}, then the route ID is added to the freezing table, the freezing counter is increased by one, its timer starts working, and all segments of the route will be labelled as *"frozen"*. Otherwise a tear down procedure is initiated and segments of the route are labelled as *"available"*.

Note that when a route is labelled as *"frozen"*, it means that it is actually available for use. In addition, frozen routes are not subject to snooping and flushing. Once a frozen route is selected by a call, its state will be changed to *"in-use"*. If at any stage, the timer for a frozen route expires and the route is not being used by a call, a tear down procedure will be initiated for the route. Then, the route will be deleted from the freezing table, and its state will be changed to *"available"*. The flowchart in Figure 2.7 details the route freezing in a cache element at a border node.

2.7.1 Freezing Period

The freezing period (T_{freeze}) defines how long a route can be kept frozen. A very large T_{freeze} can jeopardize the overall performance of the routing because it reduces the amount of available bandwidth in the network over a long period so that the network can serve fewer calls. On the other hand, with a very short T_{freeze}, new arrivals may loose the opportunity of selecting a frozen route. T_{freeze} can lead to different results under different call arrival rates and call holding times. In addition, it has to be considered in parallel with other freezing parameters.

2.7.2 Freezing Bandwidth Threshold

Route freezing can easily become a bandwidth-hungry technique. Therefore, we need to limit the amount of frozen bandwidth in the network. While the freezing period defines a limit in terms of time, the freezing bandwidth threshold (B_{freeze}) defines a limit in terms of size. In fact, B_{freeze} creates a criterion to select cached routes for freezing. Therefore, only the cached routes whose bottleneck bandwidth is less than B_{freeze} are selected for freezing. A larger value of B_{freeze} leads to the selection of wider routes. A smaller B_{freeze} leads to the selection of tighter routes.

2.7.3 Freezing Size

In addition to a freezing period and freezing bandwidth threshold, the number of frozen routes can greatly affect the performance of the routing. Freezing size (N_{freeze}) defines the maximum allowed number of simultaneously frozen routes at a border node. A very large N_{freeze} can lead to the consumption of a significant amount of bandwidth in the network and increases the blocking ratio for arriving calls. However,

new call arrivals have a greater opportunity to use the frozen routes from the cache, which provide a 100% success guarantee.

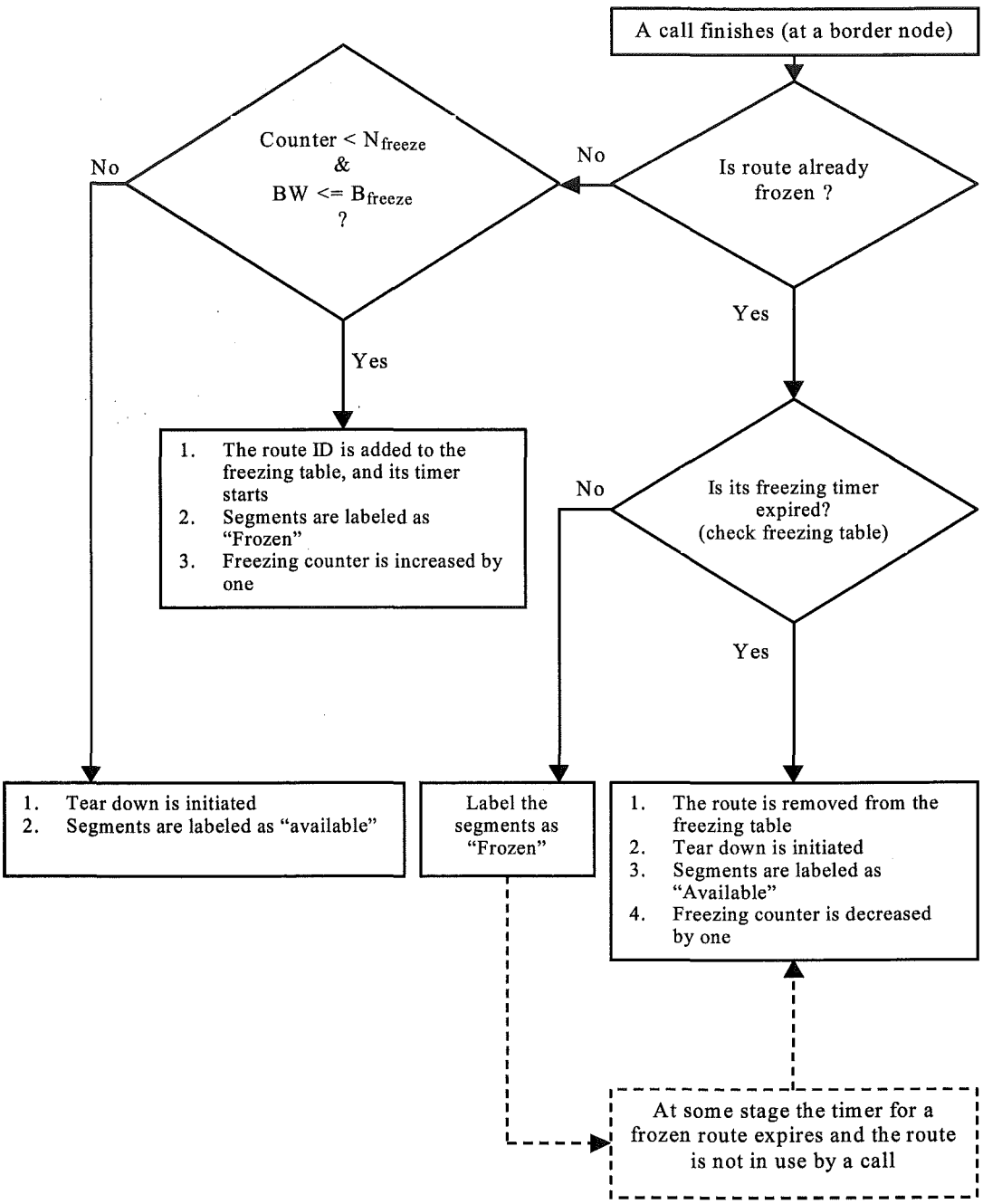
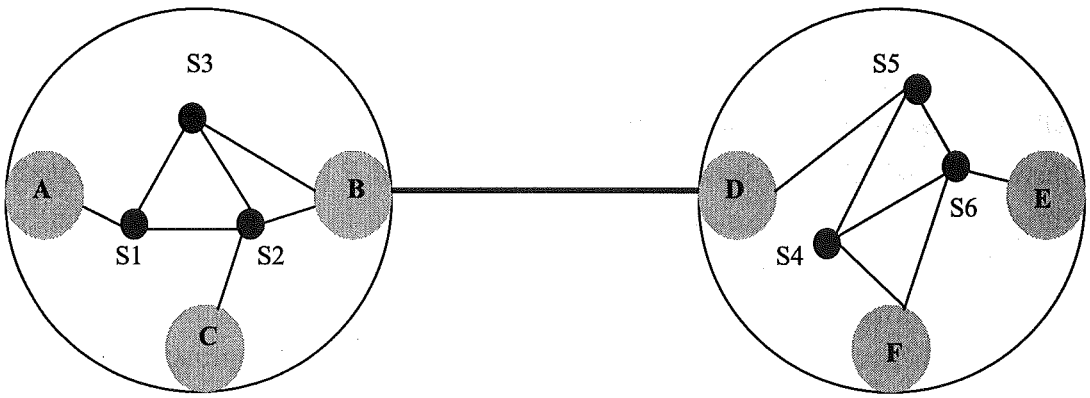


Figure 2.7: The route freezing procedure in a border node



Internal structure of the cache at the border node B at the time T, when a call arrives at the border node B and request a route to F:					
ID	Bandwidth	State	Ingress segment	Egress segment	Address list
101	12	"In-use"	A-s1-s2-B	B-D	A-B-D-E
234	10	"Available"	A-s1-s2-B	B-D	A-B-D-F
132	11	"Available"	C-s2-s3-B	B-D	C-B-D
198	3	"Available"	A-s1-s2-s3-B	B-D	A-B-D-E

The corresponding end-to-end routes for the above route IDs are as follows:
101 : A-s1-s3-B-D-s5-s6-E
234 : A-s1-s2-B-D-s5-s6-F
132 : C-s2-s3-B-D
198 : A-s1-s2-s3-B-D-s5-s4-s6-E

At the time T, when there are four cached routes in the network, a call arrives at the border node B, and requests a route to border node F with 8 megabits/second capacity.

- *The cache at the border node B is searched and such a route is not found.*
- *An on-demand route computing procedure is initiated at border node B.*
- *This happens despite the fact that route 234 is available and can satisfy the requested bandwidth. It is not selected because it can be re-used only at its terminal border nodes (A and E). This decreases the cache utilization and causes unnecessary on-demand route computing load.*
- *Route borrowing addresses this problem by allowing partial usage of end-to-end cached routes.*

Figure 2.8: A simple scenario describing route borrowing

2.8 Route Borrowing

Until now, we have assumed that the cached routes are selected by arriving calls in an end-to-end fashion. In other words, upon the arrival of a call at a border node, the cache element at that border node is searched only for routes that are started or ended

in that border node. Although simple to implement, this approach eliminates the partial usage of end-to-end cached routes. This is clearly depicted in Figure 2.8. As can be seen, the arriving call at the border node B requests a QoS route with 8 Mbits/second bottleneck bandwidth to border node F. At the same time, we see that there is a cached route between border nodes A and F that crosses border node B, satisfies the requested bandwidth by the call, and is in the *"available"* state. The arriving call, however, does not use this cached route because the route is originally computed between border nodes A and F. Consequently, a separate route has to be computed between border nodes B and F. This end-to-end approach, especially in large networks with potentially long routes, reduces the cache usage and leads to more on-demand route computing. To alleviate this, we propose a technique called *route borrowing*. Route borrowing allows a call to partially borrow an end-to-end cached route instead of computing a new one, and the whole route is labelled as *"in-use"* so that it can not be used by another call.

2.9 Reliability Considerations

Throughout this thesis, we have assumed that the cache elements are always perfectly reliable, and there is no link failure in the network. In reality, a cache element may be unable to function properly because of various hardware and software problems at the border nodes. In addition, due to link failure, the control messages, which are transmitted between cache elements for different operations, can be lost. Therefore, as far as its reliability is concerned, the distributed cache architecture by itself does not introduce extra risk. Instead, its reliability depends on the reliability of the network infrastructure. As we have discussed in Chapter 8, this issue needs thorough investigation that goes beyond the scope of this thesis.

2.10 Conclusions

In this chapter, we detailed our distributed cache architecture, which is proposed as an end-to-end scalable solution to reduce the on-demand route computing load in QoS capable networks. We have assumed a multi-domain network model, where domains are subject to topology aggregation, and have also adopted the link state source routing model, coupled with a bandwidth based QoS model. The proposed architecture can operate in association with any QoS capable routing algorithms. We briefly classify different aspects of the proposed distributed cache architecture as follows:

- Because of its *distributed* nature, it can scale to very large networks. Additionally, it has been designed to be easily deployable in networks with multiple domains.

- A novel cache content management/replacement technique called *cache flushing* has been proposed, which suits the distributed nature of the cache architecture. In contrast to the traditional cache replacement techniques that take action when the cache is full and a new entry has to be added, cache flushing works in the background and always maintains some free space in the cache elements.

- Once a route is cached, the distributed cache architecture does not rely on network state updates, and operates independently. Therefore, the cache architecture does not suffer from inaccuracy of the network state information caused by the topology aggregation, delays in the distribution of the network states, or the network state update interval. Instead, our architecture directly monitors *only those parts of the network that are more likely to be used.* In this way, it intelligently adapts to the changes in the network states. This is done by a technique called *cache snooping,* which has been developed to alleviate the effects of network state fluctuations on the cached routes. To incorporate the cache snooping in the distributed cache architecture efficiently, several parameters control the functionality of the cache snooping:

    1. The snooping interval controls the frequency of snooping in a cache element.

    2. The snooping size controls the number of segments selected each time snooping initiates in a cache element.

    3. The snooping policy outlines a set of criteria for selecting the most useful segments.

- Cache snooping increases the *routing tolerance* to inaccurate network state information. As we will study in Chapter 5, this improves the overall routing performance, especially in the presence of highly inaccurate network state information.

- We have introduced a technique called *route freezing.* Route freezing creates temporary *QoS circuits* that are completely isolated from the changes in the network states so that they offer 100% accuracy. Several parameters control the functionality of the route freezing:

1. The freezing period controls the freezing time of a route.

2. The freezing bandwidth threshold defines a criterion for selecting routes for freezing.

3. The freezing size controls the maximum number of simultaneously frozen routes selected for freezing in a cache element.

- *Route borrowing* has been proposed to allow arriving calls to use the cached routes more efficiently and to further decrease the on-demand route computing load. The principle is to partially reuse the end-to-end cached route.

Figure 2.9 presents a structured view of the proposed distributed cache architecture. It is composed of a core architecture, which is supported by snooping, freezing and borrowing modules. Each module has been developed to increase the performance of the core architecture in a different way. The core distributed cache architecture is composed of four parts:

1. Cache elements have been deployed across the border nodes of the network domains. Routes are across these cache elements in the form of multiple interconnected segments in a distributed fashion.

2. Cache flushing and its associated parameters and policies.

3. Route selection procedures and policies.

4. Route caching procedures.

The architecture has been designed in such a way that the different modules can be added in a flexible fashion. Each module enables the core architecture to reduce the on-line route computing load more efficiently. Each module does increase the complexity of the distributed cache architecture, but also increases the its performance.

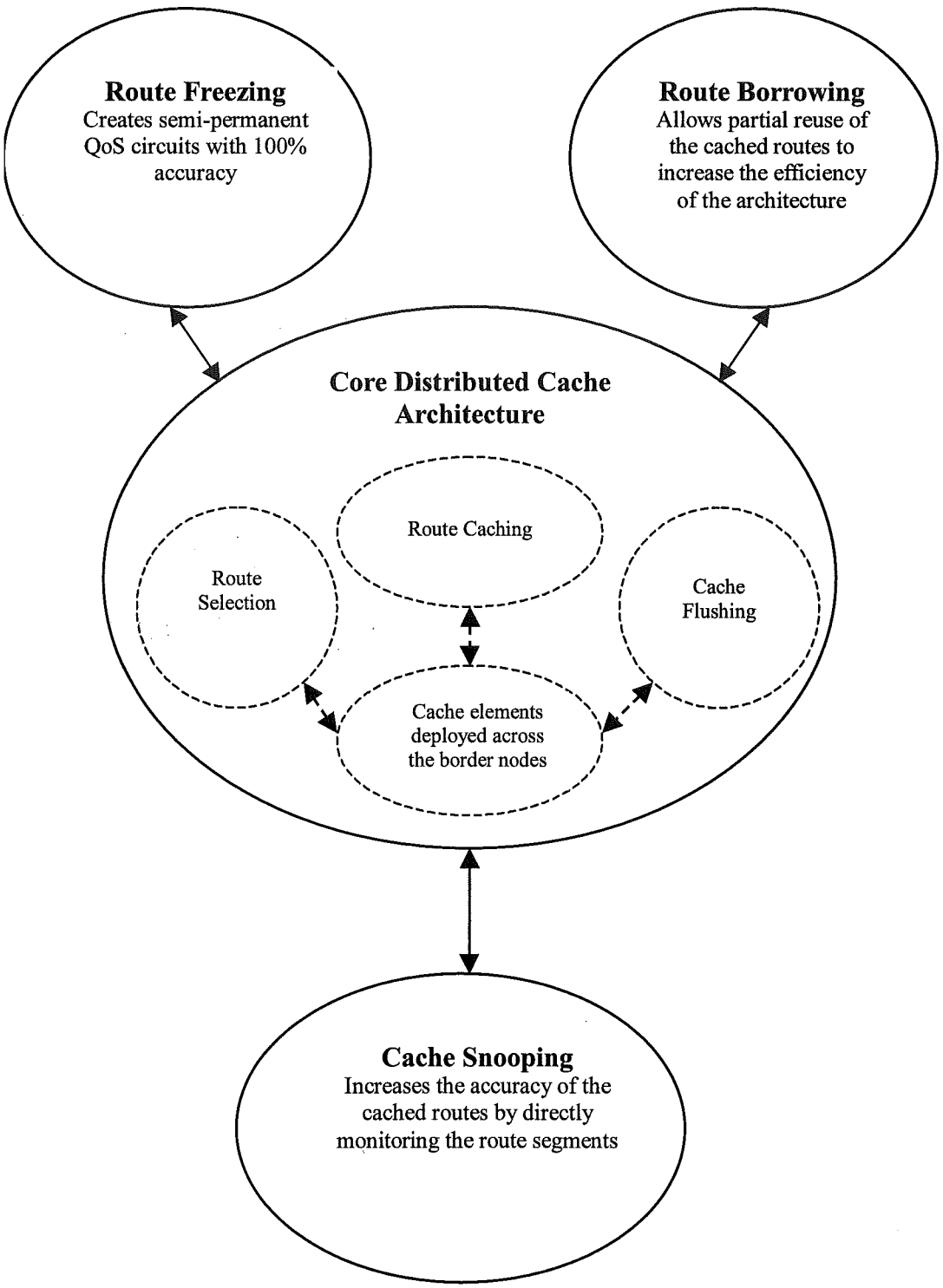


Figure 2.9: Different modules of the proposed distributed cache architecture

Chapter 3

Simulation Environment

3.1 Introduction

The evaluation of the distributed cache architecture and its associated techniques is based on an extensive set of stochastic discrete-event simulation scenarios, through which we can assess different aspects of our proposed architecture. The simulation programs have been developed in C++ and linked to AKAROA-2 [Ewi99]. AKAROA2 transparently transforms a sequential simulation program into one for parallel execution on a network of UNIX workstations, and automatically stops the simulation when the steady-state estimates of performance measures obtain the required relative precision. In this chapter, we detail major aspects of our simulation environment.

3.2 Network Topologies

Choosing adequate network topologies is a very critical factor in simulation-based performance evaluation of communication networks. This is because different routing algorithms can behave quite differently for different topologies under different network conditions such as network load or call holding time.

Throughout this thesis, we consider a multi-domain network model. As discussed in Chapter 1, such an approach is essential to achieve scalability in large communication networks. In such a model, a large network is divided into several domains. The topology and network state information inside each domain is subject to aggregation. Following this model, we have selected different network topologies at inter-domain and intra-domain levels. For inter-domain topology, we have adopted the MCI Internet backbone, which is a continent-wide real world topology used in several studies [Ma97-1, Apo98-2]. Figure 3.1 shows this topology. Each node in the topology is considered as a domain. Throughout this thesis, we use a subset of the MCI backbone to study the effect of different network sizes. For example, Figure 3.2 shows a subset of MCI topology with only ten domains. We have considered two different intra-domain topologies, which are referred to as type-1 and type-2 respectively. Figure 3.3 and Figure 3.4 show these topologies.

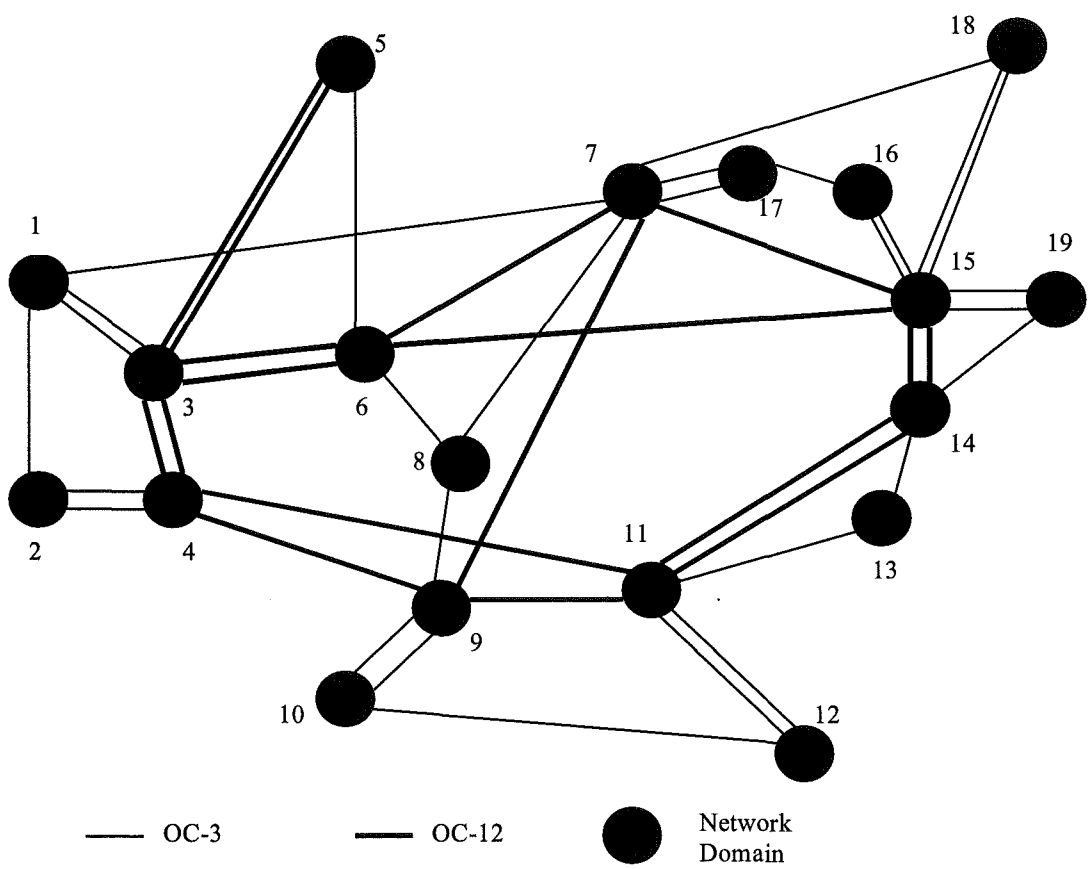


Figure 3.1: The MCI Internet backbone

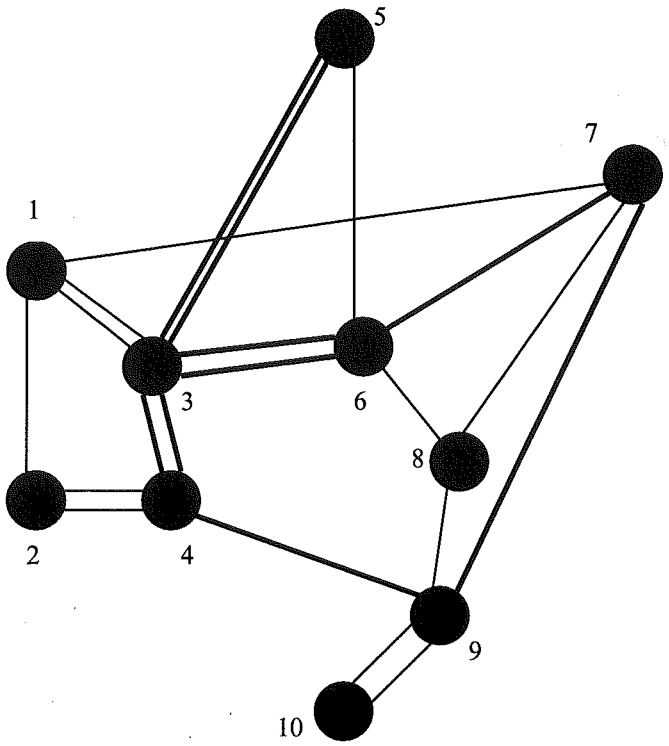


Figure 3.2: A subset of MCI Topology
with only 10 domains

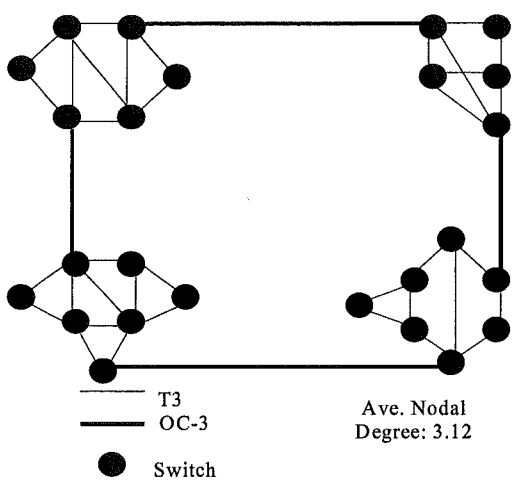


Figure 3.3: Type-1 Intra-domain
Topology

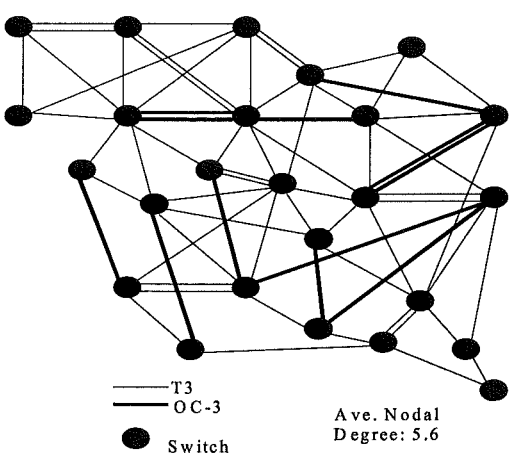


Figure 3.4: Type-2 Intra-domain
Topology

The type-1 topology resembles a multi-campus corporate network with several campus networks connected together with high bandwidth links. The type-2 topology

is an irregular topology. It is more connected and consists of more high bandwidth links. In all our simulation experiments, these intra-domain topologies are synthesized on the MCI inter-domain topology so that the internal structure of each node in the MCI topology is either a type-1 or type-2 topology. In our simulation experiments, we can select different network configurations in terms of size, type of topology, and the aggregation method.

To be clear and concise, in the remainder of this thesis we refer to the selected network configuration as Network(*d*, *t*, *aggr*), where *d* defines the number of domains in the MCI topology, *t* defines the type of intra-domain topology, and *aggr* defines the aggregation method. For example, Network(19, type-1, full-mesh) represents a network configuration as follows:

- An inter-domain topology consisting of 19 domains is used. This means the complete MCI backbone is used.
- The internal topology of each domain is a type-1 (i.e., multi-campus) network.
- A full-mesh aggregation method is used in each domain.

We have assumed all links are bi-directional. The propagation delay for intra-domain and inter-domain links is 1 and 4 milliseconds respectively.

3.3 Topology Aggregation

As discussed in Chapter 1, topology aggregation achieves scalability by hiding the detailed routing information inside a domain from the outside, so that the outside world has only an abstract view of the domain, consisting of several border nodes (also referred to as ports) and their connectivity pattern. Although topology aggregation is not the focus of this thesis, we have incorporated it into our simulation model to make our simulation scenarios as realistic as possible. We have used only two well-known aggregation mechanisms: full-mesh and symmetric star. Figure 3.5 provides a brief overview of these aggregation methods. Because of the importance of the aggregation, many researchers have developed customized software packages dedicated to studying different aggregation techniques. The work described in [Awe98-3] is an example of such work.

In the full-mesh method, the aggregated topology can be represented by a matrix with one entry per port pair. Thus a domain with N ports has a matrix of size N^2. In general, a full-mesh representation of the original topology is reasonably accurate but

does not scale very well. The symmetric star method is based on the simple and inaccurate assumption that the distances are the same between any two ports. Symmetric star is much more scalable than full-mesh but is less accurate. Figure 3.5 shows a simple view of the full-mesh and symmetric star aggregation methods. It is worth mentioning the cost metric for port-to-port distance is the same cost metric used in routing algorithms used in simulations. Therefore, the port-to-port distance under widest-shortest and shortest-widest routing algorithms is defined as the combination of the average hop count and the average available bandwidth, which are advertised separately. For competitive call and shortest-distance routing algorithms, the port-to-port distance is defined as the average available bandwidth.

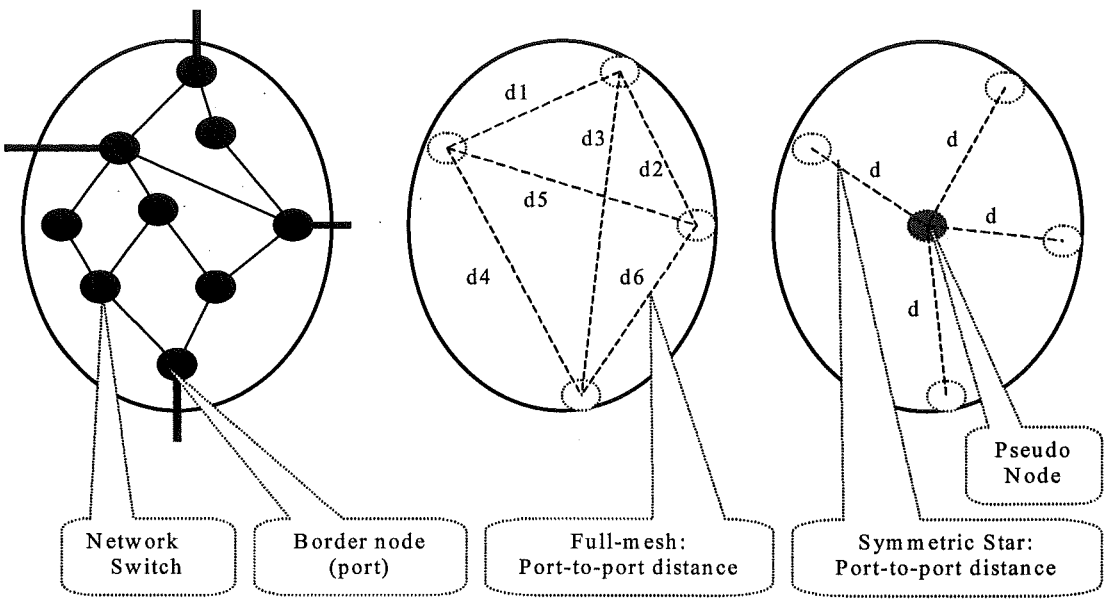


Figure 3.5: Different topology aggregation techniques
Detailed Intra-domian topology (left)
The view of the domain topology after full-mesh aggregation (middle)
The view of the domain topology after symmetric-star aggregation(right)

3.4 Network Traffic

The network traffic model is another important factor in simulation-based performance evaluation of communication networks. In this thesis, we have incorporated a comprehensive traffic model into our simulation environment as follows:

- *Call arrival process.* In all our simulation scenarios, calls arrive according to a Poisson process. In our experiments we have used mean arrival rates between 10 to 100 calls/sec.

- *Call holding time.* The exponential call holding time distribution has been used in most simulation studies of communication networks. Here we follow this trend. Throughout our experiments, we can select different mean call holding times. While the exponential distribution has been studied for voice traffic, to the best of our knowledge, the holding time distribution for video traffic is not clearly understood. For simplicity, we assume that both voice and video calls have an exponential holding time distributions. Additionally, in all simulation, we have considered the same mean call holding time for both voice and video calls.

- *Requested bandwidth.* The diversity of the requested bandwidth is an important issue that should be taken into account when designing a large QoS capable network. This is because different applications such as Internet telephony and real time video need different amounts of bandwidth. Thus we have considered a broad range of applications as follows:

  - *From 16 to 64 Kilobits/second.* This interval represents applications such as Internet telephony and videophones with various resolutions. It is assumed that the selected bandwidth in this interval. In simulation experiments, the voice calls select a value from this interval according to a uniform distribution.

  - *From 1 to 5 Megabits/second.* This interval represents applications such as streaming video, video conferencing and compressed video streams generated by multi-layered coding schemes such as MPEG-2 with various resolutions. In simulation experiments, the video calls select a value from this interval according to a uniform distribution

- *Traffic mixture.* In communication networks with heterogeneous traffic, the ratio between different types of traffic is an important factor. In accordance with our selected model for requested bandwidth, assuming λ represents the total traffic load (e.g., number of calls), λ_{voice} and λ_{video} represent the share of voice and video calls respectively, and θ represent the mixing coefficient, different traffic mixtures have been considered as follows:

$$\lambda = (\theta).\lambda_{Voice} + (1-\theta).\lambda_{Video}$$

$\theta = 0.2$ $\Rightarrow$ Video dominated traffic
$\theta = 0.50$ $\Rightarrow$ Equal voice/video traffic
$\theta = 0.8$ $\Rightarrow$ Voice dominated traffic

3.5 Routing Algorithms

As detailed in Chapter 2, we have considered a link-state source routing model similar to that proposed in the ATM PNNI standard [ATM96]. However, to gain a broad evaluation of our proposed QoS routing architecture in different circumstances, we have incorporated four different routing algorithms in our simulation environment:

- *Widest-shortest path.* This algorithm first selects all feasible routes with minimum hop counts, then selects the one with the maximum available bandwidth [Gue97-2]. We have implemented this algorithm by applying Dijkestra's algorithm [Cor90] against the network topology matrix in two steps, assuming hop count and bandwidth as cost metrics in the first and the second step, respectively. This algorithm emphasizes balancing the network load by selecting the least loaded (widest) route.

- *Shortest-widest path.* This algorithm first selects all routes with the maximum available bandwidth, then selects the one with the smallest hop count. We have implemented this algorithm in a similar manner to the widest-shortest path, reversing the cost metrics in the first and the second step. This algorithm emphasizes minimizing the resource consumption by selecting the route with the minimum hop count.

- *Competitive call routing.* This algorithm defines the cost metric for link L as follows [Hao2000]:

$$Cost = \mu^{((r(L)/b(L)) - 1)}$$

r(L): Occupied bandwidth of link L
b(L): Total bandwidth of link L
μ: Algorithm parameter

The cost of a route is defined as the sum of the link costs. It selects the minimum cost route provided it is feasible, which means its cost is less than a threshold β. In our experiments, we have set $\mu = 1000$ and $\beta = 1$, based on the experiments conducted in [Hao2000].

3.6 Performance Measures

As mentioned in Section 3.1, AKAROA-2 has been used in our simulation experiments with following considerations:

- *Confidence level and precision.* We have considered a 95% confidence level, and all results are obtained with at least 5% precision..

- *Termination criterion.* All results are obtained assuming system is in steady state and the 5% precision is achieved.

The comprehensive nature of our proposed QoS routing architecture, coupled with the broad selection of network topologies, aggregation methods, network traffic conditions, and routing algorithms, demands a careful selection of credible performance measures. We have considered the following:

- *Cache utilization ratio.* The primary purpose of route caching is reusing a computed route to reduce the route computing load. This performance measure indicates how well the cached routes are being reused by arriving calls. We define it as follows:

$$\text{Cache utilization ratio} = \frac{\text{Number of calls for which at least one feasible cached route is found}}{\text{Total number of calls}}$$

The cache utilization ratio can be used only partially to assess the performance of the distributed cache architecture because it does not take into account the success or failure of the route set-up process once a route has been found in the cache. In other words, the cache utilization ratio measures the cache's usefulness from the calls' point of view.

- *Cache hit ratio.* The accuracy of the cache contents is a very important factor that can dramatically affect the performance of the cache. This is because changes in the network states can cause cached routes to become stale and unable to satisfy their associated QoS parameter (bandwidth). This, even assuming a high cache utilization ratio, can degrade the routing performance. Therefore, to measure the accuracy of cache contents, we formulate the cache hit ratio as follows:

$$\text{Cache hit ratio} = \frac{\text{Number of calls that have been successfully established using a cached route}}{\text{Total number of calls for which at least one feasible cached route is found}}$$

- *Reduction in route computing load.* This is the production of the cache hit ratio and the cache utilization ratio. This measure shows how much reduction in route computing load is achieved by the proposed distributed cache architecture. For example, a 0.4 for this measure means that 40% of the arrivals are routed via the cache without on-demand computing. The reduction of the route computing load is defined as follows:

$$\textit{Reduction of route computing load} = \textit{Cache utilization ratio} * \textit{Cache hit ratio}$$

OR

$$\text{Reduction of route computing load} = \frac{\text{Number of calls that have been successfully established using a cached route}}{\text{Total number of calls}}$$

- *Bandwidth blocking ratio.* If all calls request the same amount of bandwidth, a lower call blocking ratio implies more QoS calls accepted into the network. However, when calls can request different amounts of bandwidth, a low call blocking ratio does not necessarily mean high efficiency. It is defined as follows: We need to make sure that the proposed distributed cache architecture does not increase the bandwidth blocking ratio.

$$\text{Bandwidth blocking ratio} = \frac{\text{Total amount of rejected bandwidth}}{\text{Total amount of requested bandwidth}}$$

3.7 Conclusions

In this chapter, we detailed the major building blocks of our simulation environment and performance measures. These building blocks have been designed with two criteria in mind:

- *Realism.* Our simulation results can provide credible insight and comment on the adequacy of different aspects of the proposed distributed cache architecture.

- *Broadness.* We can create a broad variety of conditions that can occur during the operation of a real world large network.

In accordance with the above criteria, we have designed a comprehensive simulation environment with four major building blocks as follows.

1. *Network topology.* Following our assumptions about the network model, we have considered separate topologies at inter-domain and intra-domain levels respectively. For inter-domain topology we have adopted a real continent-wide MCI backbone. For intra-domain topologies we have considered two different topologies with different degrees of connectivity and link bandwidth.

2. *Topology aggregation.* We have incorporated topology aggregation into our simulation model, because it is agreed that topology aggregation is an essential technique in large networks with scalability requirements. We have adopted two well-known symmetric-star and full-mesh aggregation techniques.

3. *Network traffic.* We have incorporated a comprehensive traffic model with special focus on the following aspects.

    - Call arrival distribution.
    - Call holding time distribution.
    - The characteristics of the requested bandwidth by calls.
    - Traffic mixture.

4. *Routing algorithms.* We have used several well-known routing algorithms to assess the behaviour of the distributed cache .

Finally, we have selected several performance measures, such as the cache utilization ratio, the cache hit ratio, and the bandwidth blocking ratio to evaluate different aspects of the proposed architecture under a variety of conditions. The diagram in Figure 3.6 shows a summary of our simulation environment.

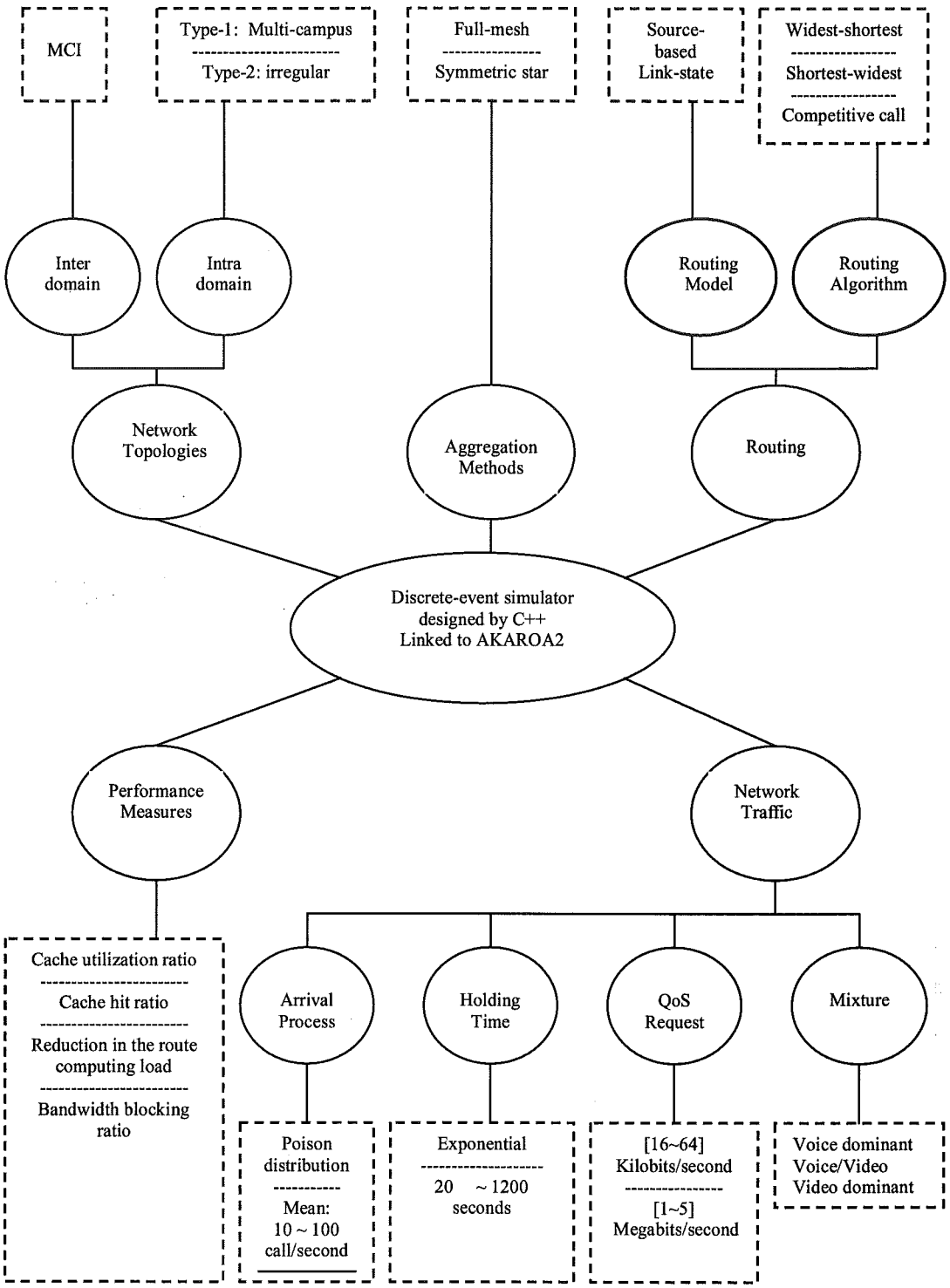


Figure 3.6: Summary of the simulation environment and assumptions

Chapter 4

Performance Evaluation of the Core Distributed Cache Architecture

4.1 Introduction

In this chapter, we evaluate the performance of the core distributed cache architecture under a variety of conditions. Techniques such as cache snooping, route freezing and route borrowing have not been considered at this stage, but will be incorporated into the core architecture in Chapters 5, 6, and 7, respectively. This approach allows us to present a well-structured and clear evaluation of these techniques and ideas. As outlined in Chapter 2, the core architecture consists of the following elements:

1. The distributed cache architecture is deployed in a multi-domain network. Under the proposed distributed cache architecture, each border node at a network domain has a cache element. An end-to-end route is stored in the form of several interconnected segments across multiple caches at the border nodes, where the route has crossed. This forms the foundation of the proposed architecture.

2. The route caching and route selection procedures and policies form the basic operations required for using the distributed cache architecture.

3. Cache flushing, and its associated parameters and policies, is cache content management technique.

All results are obtained by stochastic discrete event computer simulation, coupled with a comprehensive set of simulation assumptions. We focus on three main aspects of the core architecture: cache size, route selection policies, and cache flushing. We use the cache utilization ratio and cache the hit ratio as our main performance measures. The remainder of this chapter is organised as follows. In Section 4.2, we investigate the effects of the cache size on the cache utilization ratio under a variety of different network and traffic conditions. We attempt to identify an adequate cache size for the rest of our experiments. In Section 4.3, we investigate the effects of the route selection policies and study their interaction with other factors such as the network topology, the network traffic mixture, and the network load, to name a few. The goal is to identify those policies that help to lower the route computing load more effectively. In Section 4.4, we study the proposed cache flushing technique and its associated parameters and policies. We wish to evaluate the effectiveness of the flushing under a broad range of network and traffic conditions, and identify a balance between various flushing parameters and policies. Section 4.5 concludes this chapter.

4.2 The Size of Cache Elements

Under the proposed distributed cache architecture, by the size of the cache element we mean the number of entries in a cache element at a border node. As detailed in Chapter 2, each entry of a cache element at a border node saves all necessary information about one or two segments of an end-to-end route. Choosing a proper size for the cache elements at border nodes is important for two reasons.

1. The size of the cache elements can significantly affect the cache utilization ratio. If the cache elements are large, the distributed cache architecture can save a higher number of end-to-end routes. Consequently, the arriving calls are more likely to be routed via the distributed cache architecture instead of by on-demand route computing. Therefore, the cache utilization ratio should be maximized to reduce the route computing load more effectively (note that the reduction in route computing load is defined as the multiplication of the cache utilization ratio and the cache hit ratio).

2. The cache elements at the border nodes reside in the memory. To this end, the goal is to minimize the amount of required memory. However, it is not feasible to quantify the exact memory requirements of the cache elements at the border nodes

in the form of the number of bytes. This is because such detailed information as the length of the network addresses can vary in different technologies. In addition, the required memory to store a segment depends on the topology of the network domains.

When studying the effects of the size of cache elements, we use only the cache utilization ratio. The cache hit ratio is a measure of the accuracy of the cached routes.

4.2.1 Impact of Network Topology and Network Size

In Figure 4.1, we study the effects of the size of the cache elements on the cache utilization ratio when different intra-domain topologies have been used. The influence of the size of the inter-domain network has also been studied. As a reminder, the size of the inter-domain network is defined as the number of its domains. We can observe several phenomena as follows:

1. If the size of the inter-domain network is constant, we observe that having larger cache elements initially results in a higher cache utilization ratio up to a certain point, after which increasing the size of the cache elements has less effect on the cache utilization ratio. This is followed by a final settlement or a very small growth in the cache utilization ratio. This is because increasing the size of the cache elements initially provides the opportunity for caching a larger number of successfully computed routes so that arriving calls are more likely to find a feasible route in the distributed cache architecture. However, the total capacity of the network for accepting new calls depends on the adopted intra-domain topology and the topology aggregation.

2. Considering the same intra-domain topology, we observe that in a smaller network, the cache utilization ratio increases faster and settles with a smaller size for cache elements. However, the overall cache utilization ratio is larger than that achieved in a bigger network. This is because in a smaller network, new arrivals can choose their source and destination from a smaller *sample space*. This increases the likelihood of reusing the cached routes. In addition, assuming the same traffic conditions, a smaller network has less overall capacity so that fewer calls can be routed successfully. Therefore, the utilization ratio in a smaller network tends to settle at a smaller cache size.

3. The intra-domain topology can greatly influence the cache utilization ratio. Considering the same network size, using the type-2 topology always leads to a

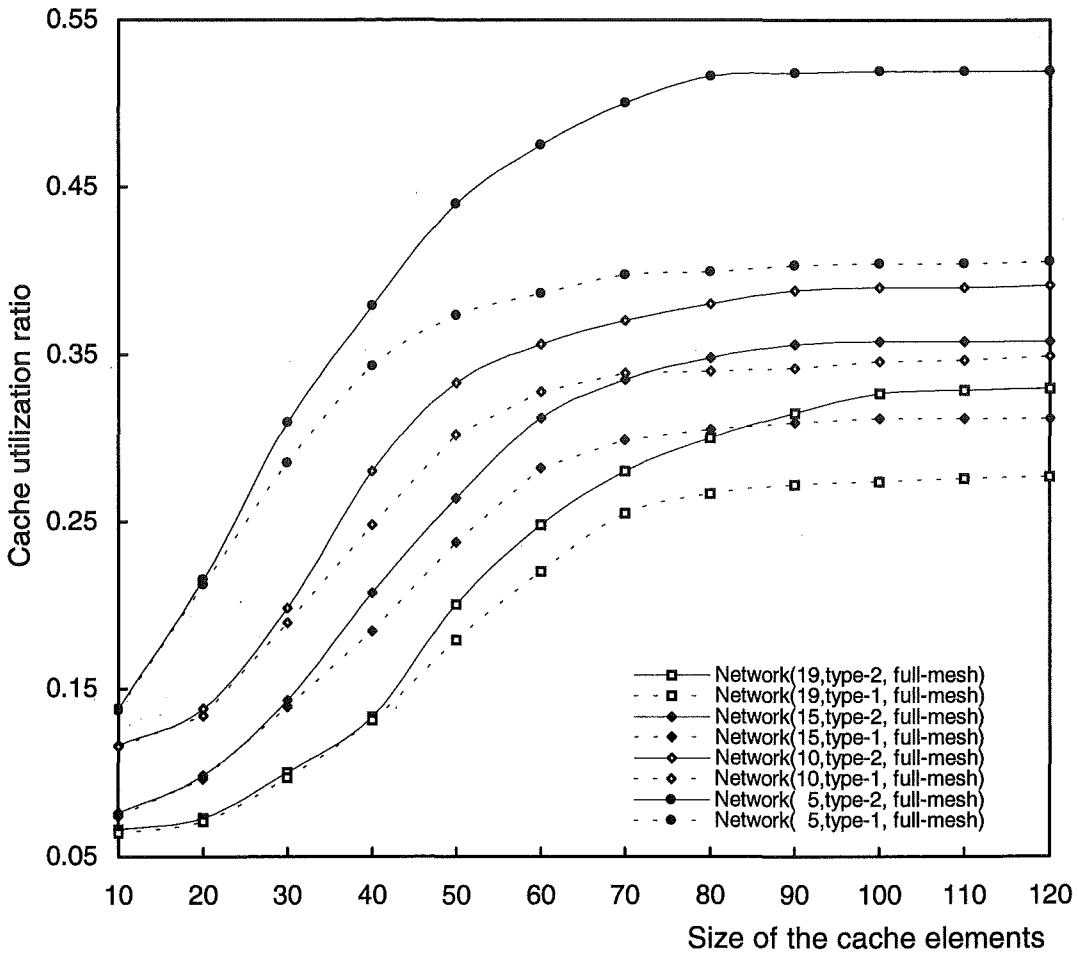


Figure 4.1: The impact of the intra-domain network topology and the network size on the cache utilization ratio
80% voice, MRC selection policy, LRC flushing policy, flushing size = 2
Mean call holding time = 3 minutes

higher utilization ratio. As stated in Chapter 2, the type-2 topology is a more connected topology. In addition, it has a greater number of high bandwidth links. Therefore, more routes can be successfully computed and stored in the distributed cache architecture. This, coupled with a higher probability of caching multiple routes between a pair of border nodes, increases the likelihood of reusing the cached routes, which leads to a higher cache utilization ratio.

4.2.2 Impact of Topology Aggregation

In Figure 4.2, we study the impact of topology aggregation on the cache utilization ratio, when different cache sizes are used. To present a clear image of this impact, we have used the type-2 topology, which is more connected and includes a greater

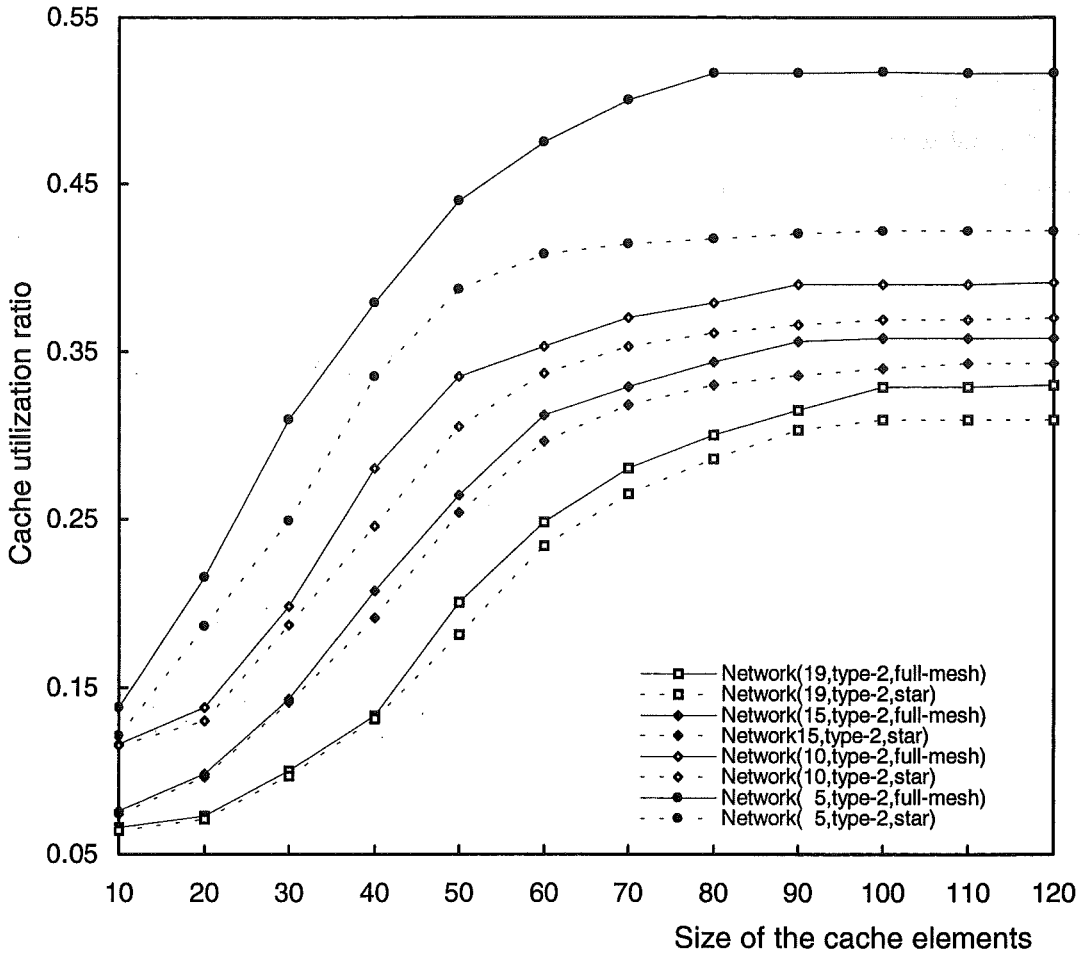


Figure 4.2: the impact of the topology aggregation on the cache utilization
80% voice, MRC route selection policy
LRC flushing policy, flushing size = 2
Mean call holding time = 3 minutes

number of high bandwidth links. We observe that for all network sizes, full-mesh aggregation results in a higher cache utilization ratio. This is because it presents a more accurate image of the internal topology and the network state information of the domains. Therefore, more routes can be computed successfully and stored in the cache. In addition, the higher accuracy of the internal topology of the network domains increases the chance of computing multiple routes between a pair of border nodes. For these reasons, the likelihood of reusing the cached routes is increased, which results in a higher cache utilization ratio. In addition, we observe that in a smaller network, topology aggregation makes a clear difference in the cache utilization ratio even with a small cache size. However, this is not the case for larger networks, where the difference is clear only at larger cache sizes. The reason is that a smaller network presents a smaller sample space for arrivals to choose from, so,

arrivals are more likely to reuse the cached routes even when the cache size is small. Finally, we observe that when star topology aggregation is used, the cache utilization ratio settles at a smaller cache size. This is because star aggregation offers less accurate network state and topology information. This leads to a lower routing performance in terms of the number of accepted calls into the network. Therefore, a smaller cache size would suffice to achieve the ultimate utilization ratio.

Comparing results in Sections 4.2.1 and 4.2.2, we can conclude that the aggregation techniques and the adopted network topologies influence the cache utilization ratio in similar ways. The reason for this similarity is quite straightforward. Both aggregation and the network topology directly impact on the quality of the routing decisions and consequently the number of successfully computed routes. For example, assuming the same network size, aggregation, and traffic conditions, using the type-2 intra-domain topology results in a higher cache utilization ratio than that achieved with type-1 topology. Similarly, assuming the same network size, topology, and traffic conditions, using the full-mesh aggregation results in a higher cache utilization ratio than that achieved using the star aggregation. In both cases, the reason for the higher cache utilization ratio is the higher number of successfully computed and cached routes, coupled with the higher probability of computing multiple routes between a given pair of border nodes.

4.2.3 Impact of Call Arrival Rate

As outlined in Chapter 2, in our simulation model, the network traffic is controlled by the call arrival rate, the mean call holding time, and the traffic mixture. We study the impacts of each factor on the cache utilization ratio. Figure 4.3 shows the impact of the mean arrival rate on the cache utilization ratio for different cache sizes. One can observe several phenomena.

1. In general, for all network sizes, a higher mean arrival rate decreases the cache utilization ratio. This happens for several reasons. Firstly, with a higher mean arrival rate, the cached routes are more frequently in *"in-use"* states. Secondly, increasing the mean arrival rate causes more bandwidth consumption and more fluctuation in the network state information so that the call blocking ratio increases. This happens because of the lack of accuracy in the network state information, coupled with the general lack of bandwidth across the network links. Consequently, fewer routes can be successfully computed and stored in the cache.

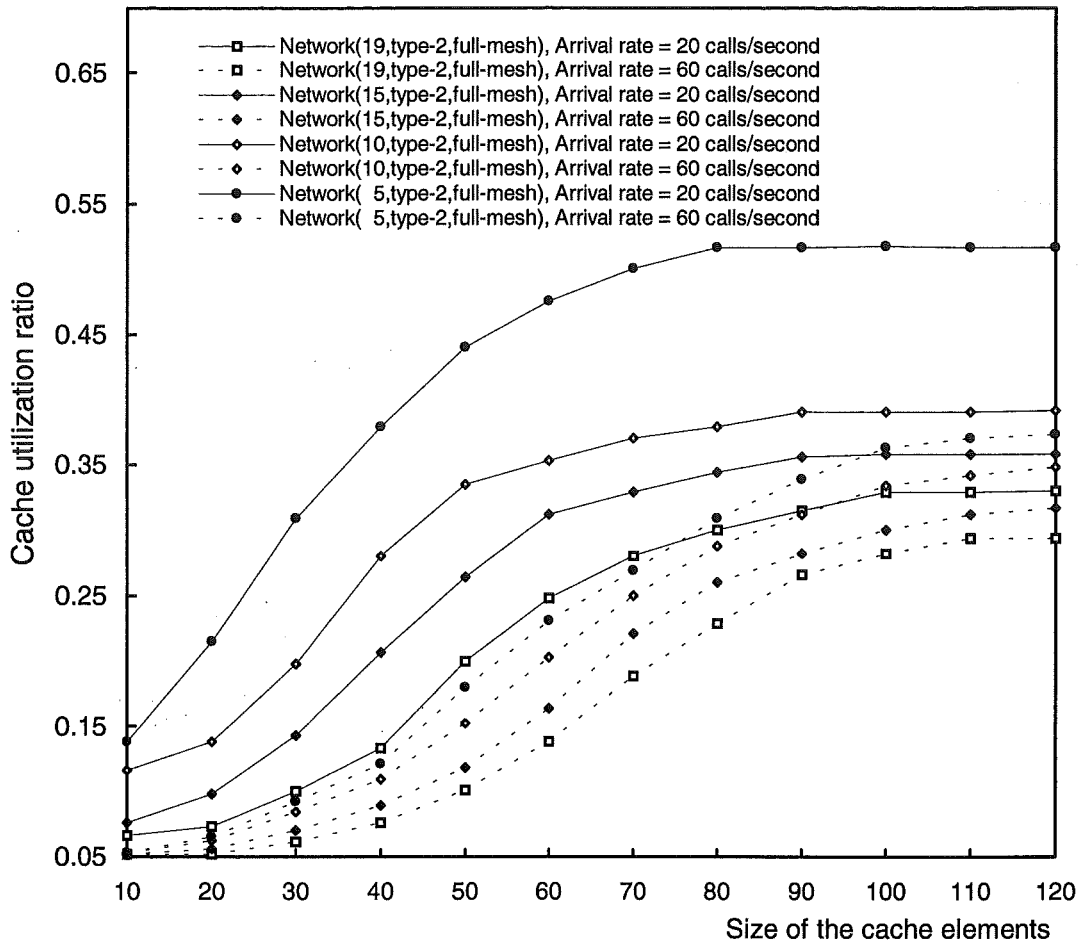


Figure 4.3: The impact of the call arrival rate on the cache utilization ratio
80% voice, MRC route selection policy
LRC flushing policy, flushing size = 2
Mean holding time = 3 minutes

2. Comparatively speaking, in a smaller network, the cache utilization ratio drops more significantly. This is because, assuming the same mean arrival rate, a smaller network has less total capacity (i.e., fewer links) so that the call blocking ratio drops more significantly by increasing the mean arrival rate. Therefore, there will be fewer cached routes to be reused.

3. We observe that for all network sizes, a higher mean arrival rate causes the cache utilisation ratio to settle at a larger cache size. This is because a higher mean arrival rate decreases the probability of the cached routes being in the *"available"* state. A larger cache size facilitates the storage of more routes. With a higher number of cached routes, one is more likely to be in the *"available"* state.

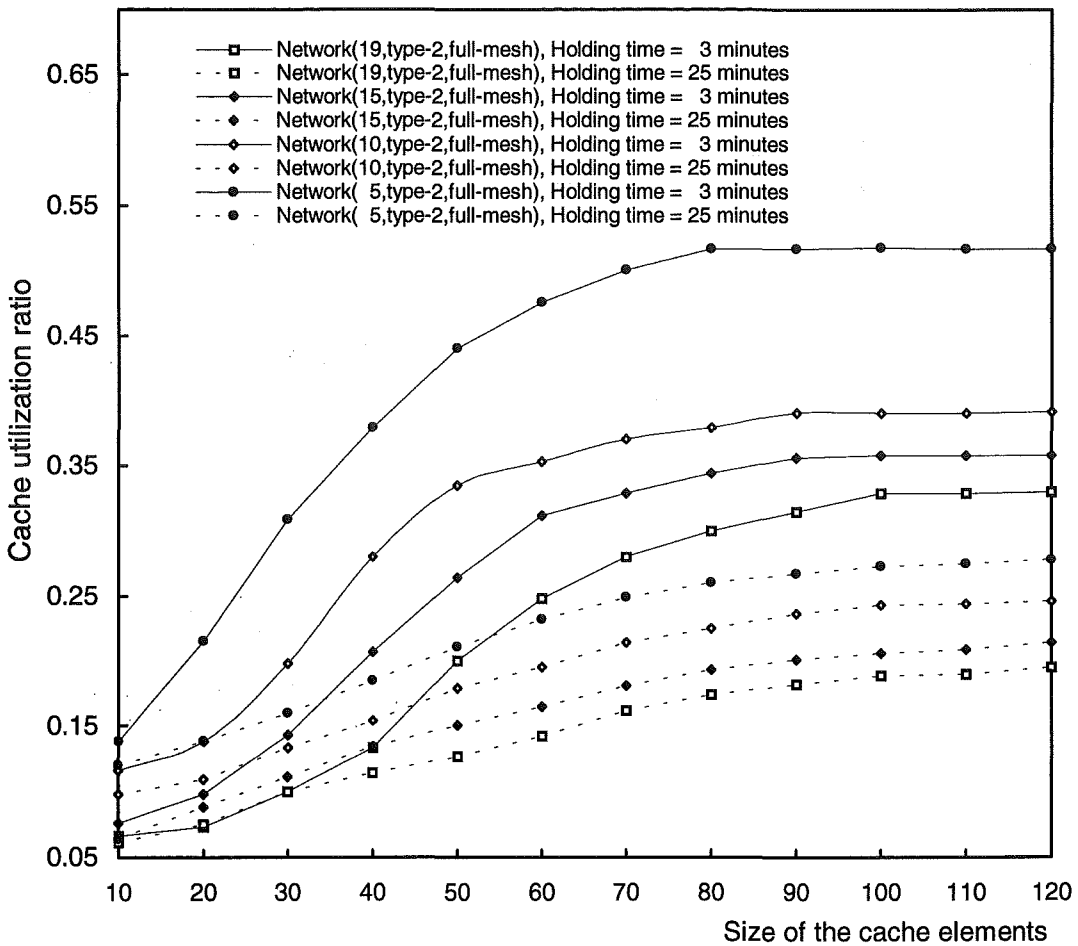


Figure 4.4: The impact of the mean call holding time on the cache utilization ratio
80% voice, MRC route selection policy
LRC flushing policy, flushing size = 2

4. For all network sizes, with a higher mean arrival rate, when the cache size is reasonably small (e.g., less than 30), we observe that the cache utilization ratio is almost close to 0. Again the reason is that a very small cache size significantly increases the probability of the cached routes being in the *"in-use"* state. In addition, a very small cache size leaves only a minimal chance for arriving calls to find a feasible route in the cache.

4.2.4 Impact of Call Holding Time

In Figure 4.4 we study the impact of the mean call holding time on the cache utilization ratio for different cache sizes. Note that we have assumed the same distribution and mean holding time for both voice and video calls.

1. For all network sizes, a longer mean call holding time reduces the cache utilization ratio. However, in contrast to the call mean arrival rate, a longer mean call holding time reduces the network state fluctuations. The problem with a longer mean call holding time is the reduction of available bandwidth across network links over a longer time. This can significantly increase the call blocking ratio, which results in fewer cached routes and consequently a lower cache utilization ratio. In such a situation, increasing the cache size will not help because the high blocking ratio does not allow the added cache size to be utilised. This is clearly reflected in Figure 4.4, where we observe that the cache utilization ratio achieved under a longer mean call holding time is reasonably insensitive to the cache size.

2. Under a longer mean call holding time, the reduction in the cache utilization ratio in a smaller network is more significant than that achieved in a larger network. This is because a smaller network has less capacity, and a longer mean call holding time can limit the available bandwidth on the network links more severely.

4.2.5 Impact of Traffic Mixture

As mentioned in Chapter 2, in communication networks with heterogeneous traffic, the ratio between different types of traffic is an important factor. We refer to this ratio as the traffic mixture. The traffic mixture defines the percentage of the voice and the video calls respectively. Because voice and video calls request very different amounts of bandwidth, the traffic mixture can significantly influence different aspects of the distributed cache architecture and the QoS as a whole. In addition, by taking the traffic mixture into account, we can evaluate the performance of the distributed cache architecture in a more realistic fashion. In Figure 4.5, we study the impact of the network traffic mixture on the cache utilization ratio for different cache sizes. In all previous experiments, we have used a fair traffic mixture (i.e., 50% voice, 50% video). Here, we use voice-dominated traffic (i.e., 80% voice, 20% video) as well as a fair traffic mixture. We observe several interesting phenomena.

1. For all network sizes, voice-dominated traffic leads to a higher cache utilization ratio. Note that voice calls request much smaller amounts of bandwidth (between 16 and 64 kilobits/second). This has two effects. Firstly, under voice-dominated traffic, more routes can be successfully computed and stored in the cache. Secondly, the likelihood of computing multiple routes between a given pair of border nodes is increased. Therefore, a higher cache utilization ratio is achieved.

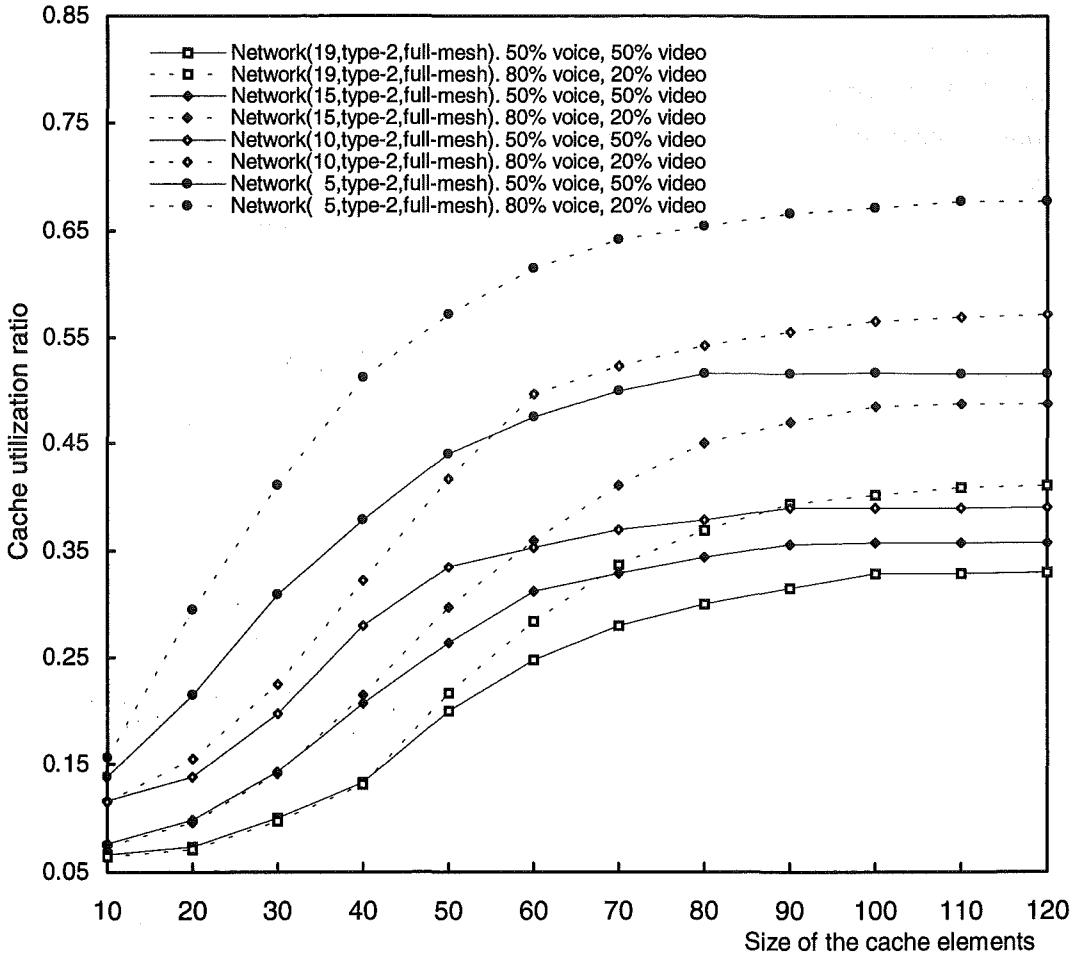


Figure 4.5: The impact of the traffic mixture on the cache utilization ratio
MRC route selection policy
LRC flushing policy, flushing size = 2
Mean holding time = 3 minutes, Arrival rate = 20 calls/second

2. Under voice-dominated traffic, the growth of the cache utilization ratio is more sensitive to the cache size. In addition, for all network sizes, the cache hit ratio settles at a larger cache size. Similarly to the previous case, this is because of the higher number of successfully computed and cached routes. With a larger cache size, the distributed cache architecture can store more routes, which leads to a higher cache utilization ratio.

4.2.6 Conclusions about the Cache Size

The experiments conducted under a variety of network and traffic conditions in Sections 4.2.1 to 4.2.5 highlight that the cache size plays an influential role in the proposed distributed cache architecture. We observe that a small cache size always

leads to a relatively low cache utilization ratio. On the other hand, we continuously observe a limit for the cache size, after which, the cache utilization does not increase, or completely settles. Although this limit changes under different conditions, we can observe it falls somewhere between 70 to 100. We have selected a cache size of 100 for the rest of this thesis. This will let us get the best out of the cache utilization ratio, while minimizing the memory consumption at the border nodes.

4.3 Route Selection Policies

As detailed in Chapter 2, under our distributed cache architecture, upon arrival of a call at a border node, the cache element of the border node is first consulted for the availability of one or more feasible routes. If more than one feasible route is found in the cache, one of them should be selected to go through the setup procedure. In such a situation, we use one of the following route selection policies:

- *Most Recently Computed (MRC).* This policy aims to minimize the probability of selecting an obsolete cached route by choosing the *youngest* route. The principle is that the most recently computed route has been subject to less fluctuation in the network states.

- *Least Frequently Used (LFU).* This policy attempts to choose the *least popular* route, therefore increasing the availability of more popular routes for future calls.

- *Widest.* This policy chooses the widest of all feasible routes. Its goal is to balance the network load. It also attempts to reduce bandwidth fragmentation.

- *Tightest.* This policy attempts to choose the best fit. It leaves wider routes for future calls with possibly higher bandwidth requirements.

After a route is selected based on the adopted route selection policy, it goes through the setup procedure. In this section, we study the impact of these policies on the cache hit ratio. We also look at their effectiveness in reducing the on-demand route computing load, which is a primary goal for the distributed cache architecture. In addition, we study their interaction with other important aspects of a QoS-capable network. Note that the reduction of the route computing load is defined as the product of the cache hit ratio and the cache utilization ratio.

4.3.1 Route Computing Load vs. Route Selection Policies

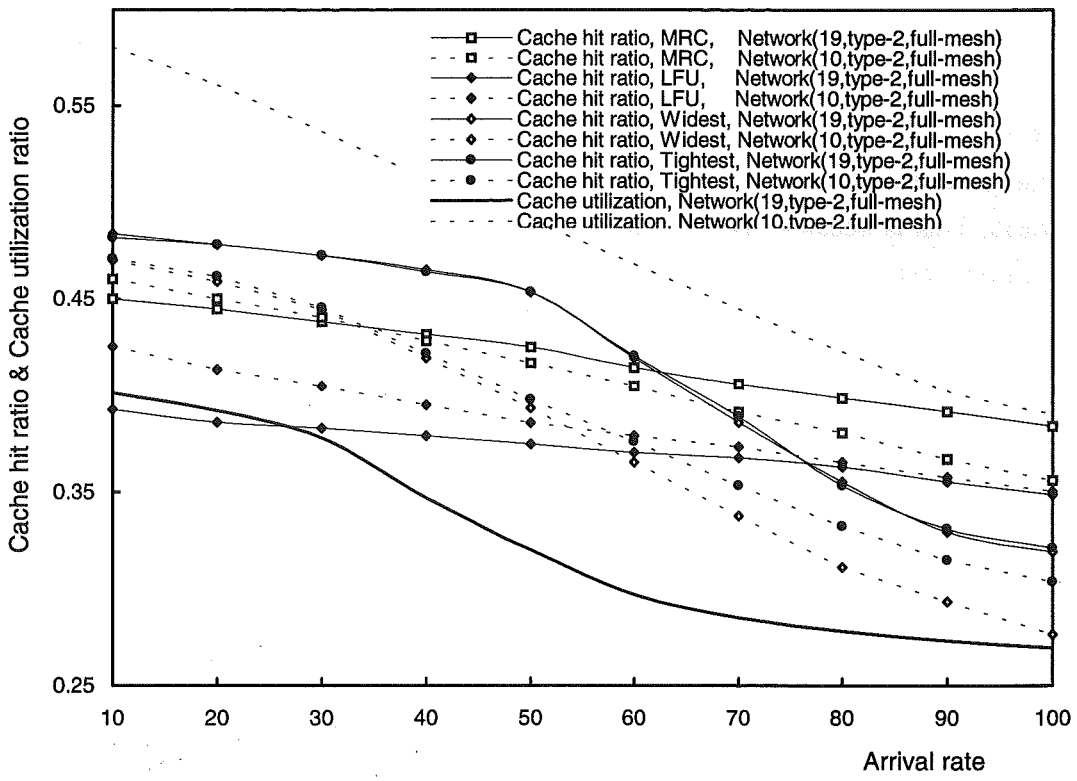
The primary goal of the proposed distributed cache architecture is to reduce the route computing load. In Figure 4.6-a and 4.6-b we investigate the interactions between the route selection policies and the reduction in the route computing load under different arrival rates and network sizes. Figure 4.6-a presents the cache hit ratio and the cache utilization ratio simultaneously, while Figure 4.6-b shows the overall reduction in the route computing load. We classify our observations as follows.

1. The route selection policies do not affect the cache utilization ratio. This is an expected result because upon arrival of a call at a border node, a selection policy is enforced only if multiple feasible routes are *found* in the cache.

2. For all network sizes and route selection policies, the hit ratio reduces by increasing the call arrival rate, although this occurs at different rates for different policies. The reason for the reduction of the cache hit ratio is the greater changes in the network states, which causes the cached routes to become obsolete faster and more frequently.

3. We observe that in all network sizes, the MRC route selection policy causes less reduction in the cache hit ratio at higher arrival rates. This is because it attempts to select the most recently computed cached routes. Therefore, MRC is shown to be the most effective policy, especially at higher loads, when the network states change rapidly. We observe that at higher arrival rates, MRC leads to a lower cache hit ratio in a smaller network. Assuming the same traffic conditions and arrival rate, a smaller network faces more state fluctuations. Consequently, MRC performs slightly worse.

4. The widest and the tightest route selection policy leads to a similar cache hit ratio. The achieved cache hit ratio under these policies, drops significantly at higher arrival rates. However, in a smaller network, the tightest policy performs slightly better at higher arrival rates. However, this experiment was with voice-dominated traffic. We also observe that both policies achieve a higher cache hit ratio in a larger network. This is because, assuming the same traffic conditions, a larger network faces less fluctuation than a smaller network. Therefore, the accuracy of the cached routes is higher.



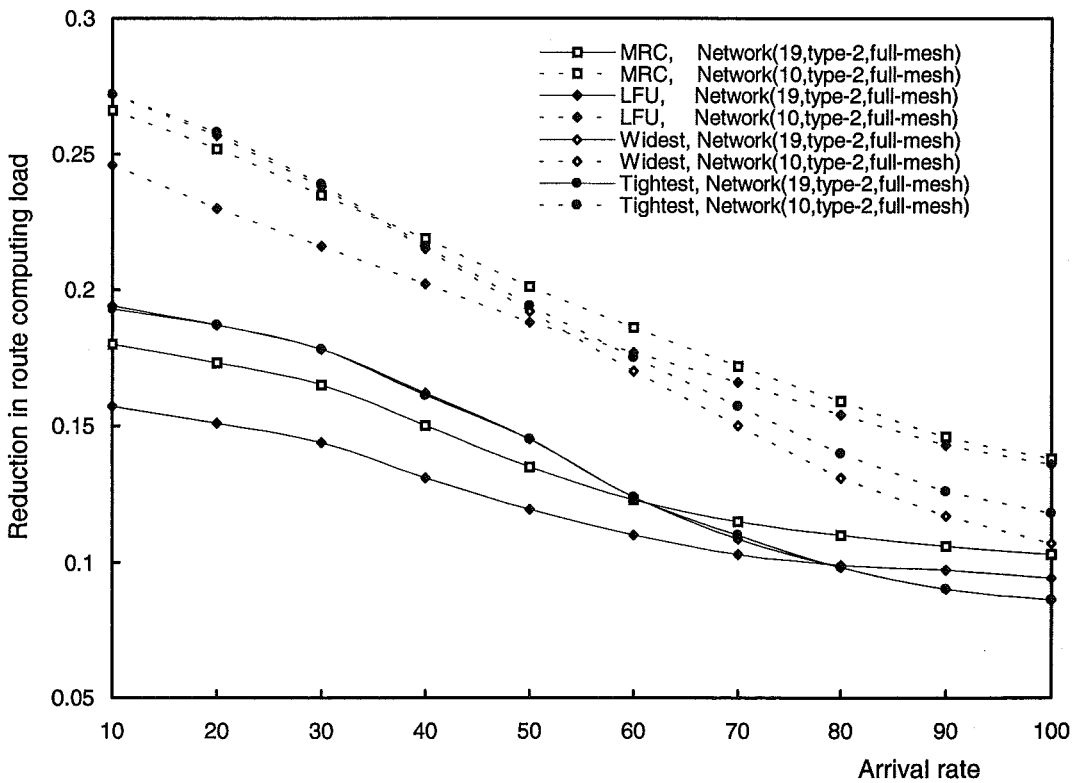


Figure 4.6-a: Impact of the route selection policies on the cache hit ratio and the cache utilization ratio

Figure 4.6-b: Impact of the route selection policies on the reduction in the route computing load.

(80% voice, LRC flushing policy, flushing size = 2)

5. Independently of the network size, the LFU route selection policy performs worse than other polices at lower arrival rates. However, under this policy, the cache hit ratio is not significantly reduced at higher arrival rates. In addition, in contrast to the other route selection policies, a smaller network leads to a higher cache hit ratio. This is because the LFU policy attempts to use those routes that have been reused least often. Because of a smaller sample space (i.e., the number of border nodes from which a call chooses its source and destination), a smaller network increases the likelihood of reusing the cached routes.

6. Figure 4.6-b shows the overall reduction in the route computing load. Table 4.1 also presents a simplified set of data corresponding to Figure 4.6-b. The reduction in route computing load means the percentage of the calls that are routed via the cache instead of on-demand route computing. As Table 4.1 shows, in a network with nineteen domains (i.e., the complete MCI backbone), assuming the MRC selection policy, the core distributed cache architecture reduces the route computing load by about 18% to 10.3%, depending on the arrival rate. At the same time, in a network with ten domains, using the MRC policy causes a 26.6% to 13.8% reduction in the route computing load, depending on the call arrival rate.

Route Selection Policy	***10 calls/sec***	***50 calls/sec***	***100 call/sec***
MRC	18%	13.5%	10.3%
LFU	15.7%	11.9%	9.4%
Widest	19.3%	14.5%	8.6%
Tightest	19.3%	14.5%	8.6%

(a): Network with 19 domains

Route Selection Policy	***10 calls/sec***	***50 calls/sec***	***100 call/sec***
MRC	26.6%	20.12%	13.8%
LFU	24.6%	18.8%	13.6%
Widest	27.2%	19.2%	10.7%
Tightest	27.2%	19.4%	11.8%

(b): Network with 10 domains

Table 4.1: The relationship between route selection policy and the reduction in the route computing load under different network sizes

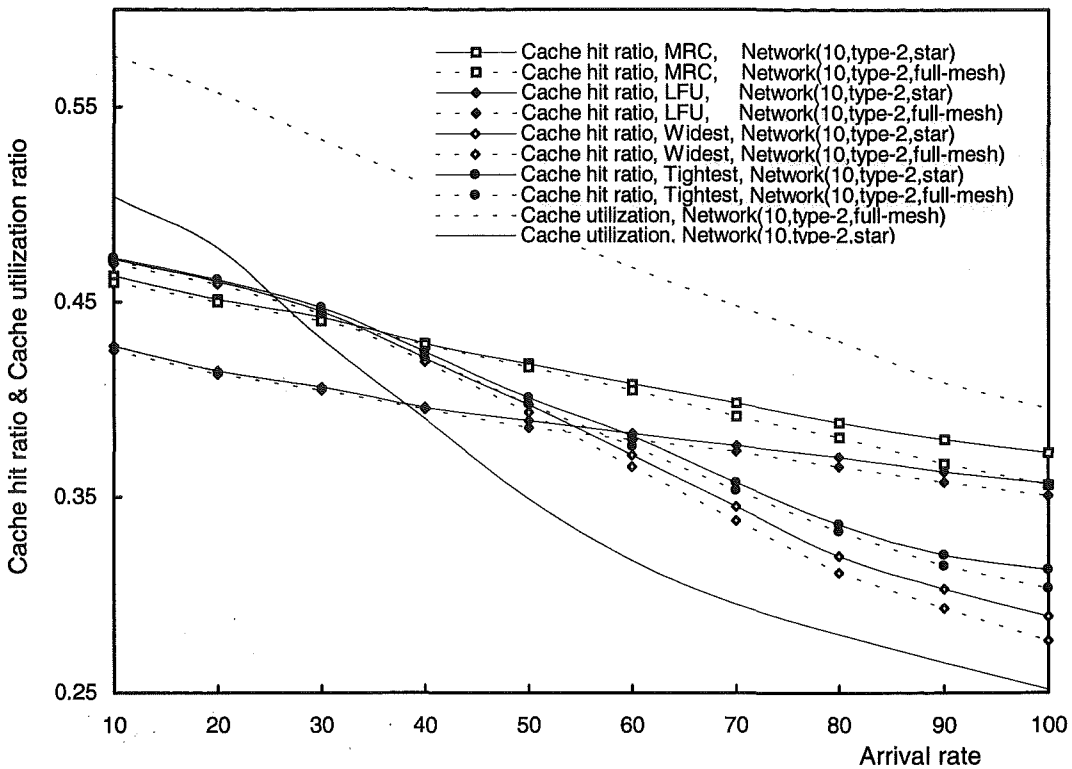
4.3.2 Impact of Topology Aggregation

As outlined in Chapter 1, topology aggregation can adversely affect the performance of a QoS-capable routing architecture because it reduces the accuracy of the network state information. We investigate the interaction between different aggregation techniques and the reduction in the route computing load under the distributed cache architecture. Figure 4.7-a presents the cache hit ratio and the cache utilization ratio simultaneously, while Figure 4.7-b shows the overall reduction in the route computing load under different topology aggregation techniques. We classify our main results as follows.

1. As a general observation, topology aggregation significantly affects the cache utilization ratio. This is because topology aggregation reduces the accuracy of the network state information by presenting only a compact view of each network domain to the rest of the network. Therefore, fewer routes can be successfully computed and stored in the cache. This lowers the cache utilization ratio because arriving calls are less likely to find a feasible route in the caches at the border nodes. As we observe, the cache utilization ratio drops significantly at higher arrival rates, when the network state information at the border nodes becomes even less accurate as a result of greater fluctuation in the network states.

2. For all arrival rates, full-mesh aggregation leads to a substantially higher cache utilization ratio than the star aggregation, full-mesh offers a more accurate image of the internal topology and state of the network domains so that more routes can be successfully computed and stored in the distributed cache architecture. This increases the cache utilization ratio. According to results shown in Figure 4.7-a, under the full-mesh aggregation, depending on the call arrival rate, cache utilization ratio varies between 58% and 40%. On the other hand, under the star aggregation, the cache utilization ratio varies between 50% and 25%, depending on the call arrival rate.

3. We observe that the choice of aggregation technique has little or no effect on the cache hit ratio. This is because the cache hit ratio is a measure of the accuracy of the cached routes, which is not affected by the aggregation technique.



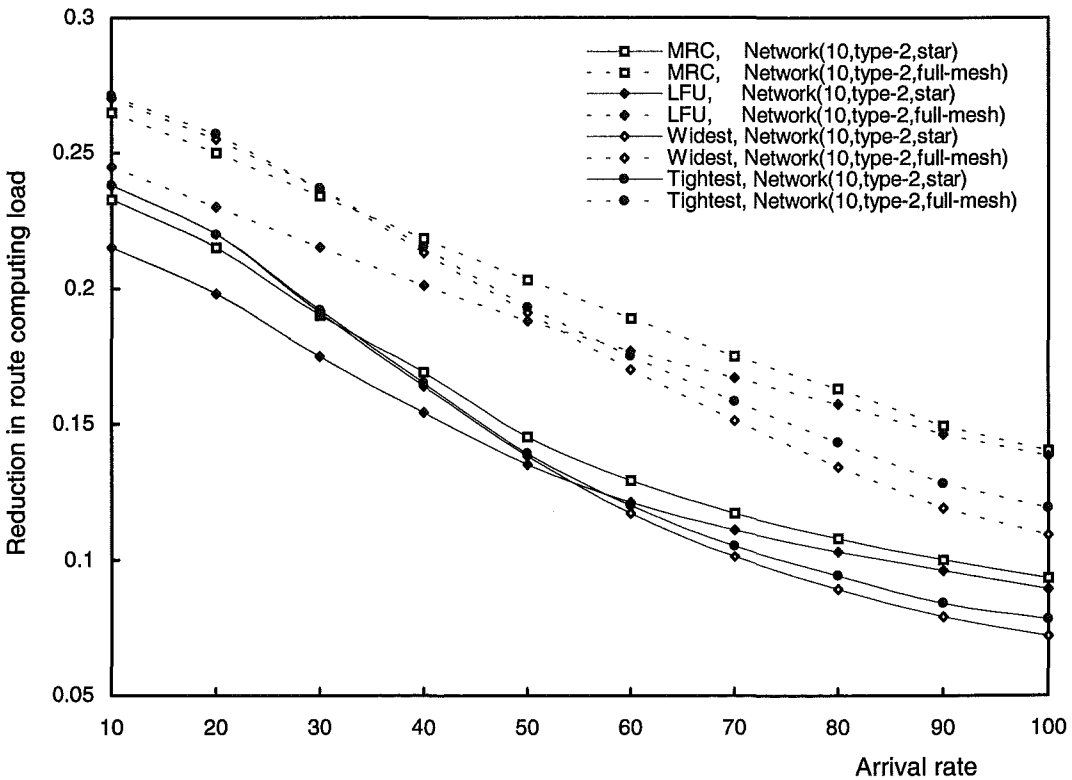


Figure 4.7-a: Impact of the aggregation techniques on the cache hit ratio and the cache utilization ratio

Figure 4.7-b: Impact of the aggregation techniques on the reduction of the route computing load

(80% voice, LRC flushing policy, flushing size = 2)

4. Figure 4.7-b presents the reduction in the route computing load under different topology aggregation techniques. Table 4.2 also presents a brief set of data from Figure 4.7-b. Under full-mesh aggregation, the route computing load has been reduced to between 27.1% and 11.9, depending on the arrival rate and the selection policy. At the same time, under the star aggregation, a 23.8% to 7.2% reduction in the route computing load has been achieved.

Route Selection Policy	***10 calls/sec***	***50 calls/sec***	***100 call/sec***
MRC	26.5%	20.3%	14%
LFU	24.5%	18.8%	13.8%
Widest	27%	19.3%	10.9%
Tightest	27.1%	19.3%	11.9%

(a): Full-mesh aggregation

Route Selection Policy	***10 calls/sec***	***50 calls/sec***	***100 call/sec***
MRC	23.3%	14.5%	9.3%
LFU	21.5%	13.5%	8.9%
Widest	23.8%	13.9%	7.2%
Tightest	23.8%	13.9%	7.8%

(b): Star aggregation

Table 4.2: The relationship between route selection policy and the reduction in the route computing load under different topology aggregation techniques

4.3.3 Impact of Traffic Mixture

In this section, we study the interaction between the traffic mixture and the reduction of the route computing load under the distributed cache architecture. By changing the ratio between voice and video calls, we create two types of traffic: voice-dominated and video-dominated. Figure 4.8-a and Figure 4.8-b present the results obtained under different traffic mixtures. We explain them as follows:

1. Video-dominated traffic leads to a significantly lower cache utilization ratio. We observe that under voice-dominated traffic, the distributed cache architecture achieves a cache utilization ratio between 58% and 40%, depending on the arrival rate, while under video-dominated traffic, the cache utilization ratio ranges between 44% and 16%, depending on the arrival rate. This is because under he video-dominant traffic, most calls request high amounts of bandwidth so that

fewer routes can be successfully computed and saved in the cache. Consequently, arriving calls are less likely to find a feasible route in the cache. This explains the significant difference between the cache utilization ratio achieved under different traffic mixtures.

2. The traffic mixture also affects the cache hit ratio, especially at higher arrival rates. We observe that under both video-dominated and voice-dominated traffic, the MRC route selection policy leads to the highest cache hit ratio. This is because the MRC policy is the most sensitive policy to the network state fluctuations (i.e., it attempts to select the cached routes with minimal exposure to the network state fluctuations). Video-dominated traffic causes larger changes in the network states. Therefore, the cache hit ratio gained under the MRC policy is significantly better than those achieved using other policies. Under voice-dominated traffic, the tightest policy achieves the second best cache hit ratio, while under the video-dominated traffic, the widest policy achieves the second best cache hit ratio. This is because under voice-dominated traffic, most cache routes are reasonably tight, while under video-dominated traffic most cache routes are wide routes (i.e., computed for video sessions).

3. According to Figure 4.8-b, and briefly shown in Table 4-3, under the voice-dominated traffic, the route computing load has been reduced to between 26.7% and 10.9%, depending on the arrival rate and the adopted route selection policy. Assuming the same conditions, under video-dominated traffic, the route computing load has been reduced to between 18% and 2.4%.

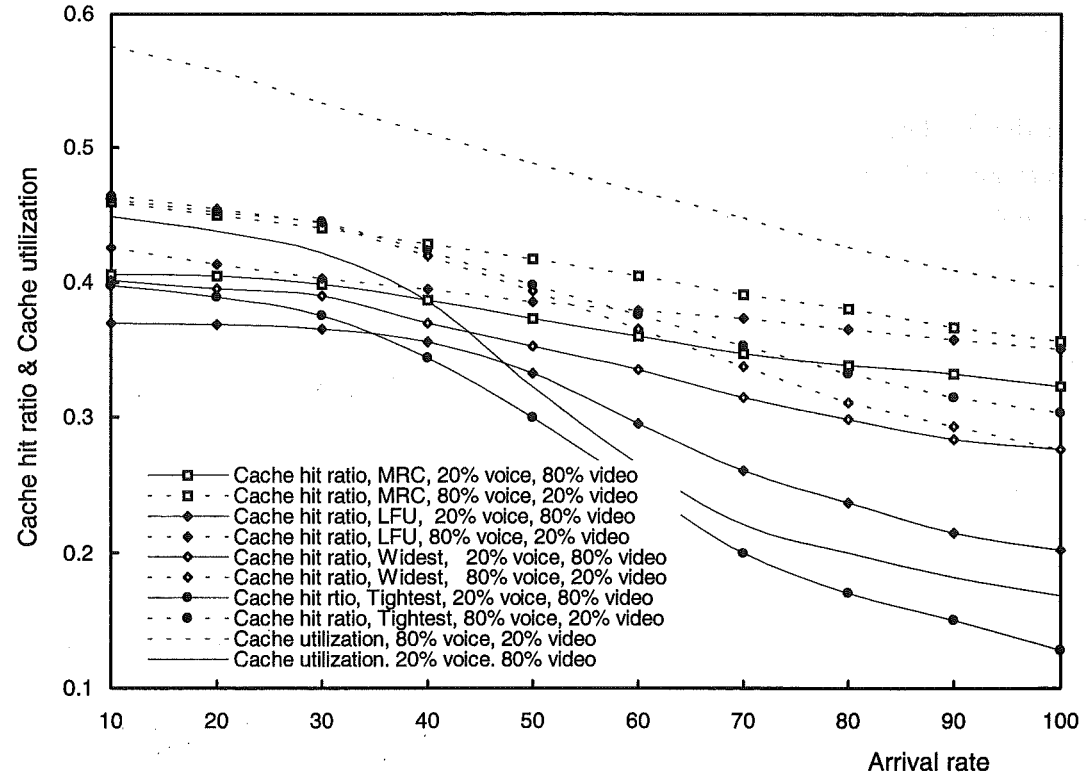
Route Selection Policy	***10 calls/sec***	***50 calls/sec***	***100 call/sec***
MRC	26.7%	20.3%	14.1%
LFU	24.5%	18.8%	13.8%
Widest	26.7%	19.1%	10.9%
Tightest	26.5%	19.4%	12%

(a): Voice-dominated traffic

Route Selection Policy	***10 calls/sec***	***50 calls/sec***	***100 call/sec***
MRC	18%	12%	6.1%
LFU	16.5%	10.7%	3.7%
Widest	18%	11.3%	5.2%
Tightest	17.8%	9.6%	2.4%

(b): Video-dominated traffic

Table 4.3: The impact of route selection policies on the reduction of route computing load under different traffic mixtures



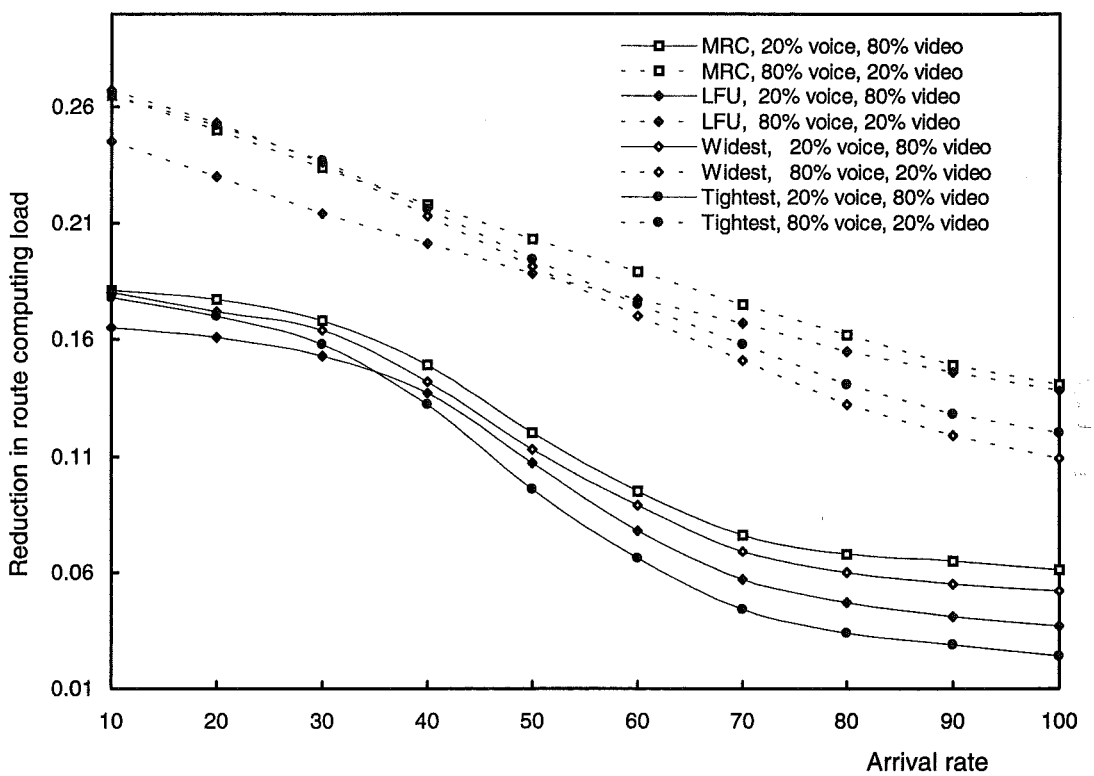


Figure 4.8-a: Impact of traffic mixture on the cache hit ratio and the cache utilization ratio

Figure 4.8-b: Impact of traffic mixture on reduction of the route computing load

Network(10, type-2, full-mesh), LRC flushing policy, flushing size = 2

4.3.4 Impact of Routing Algorithms

In this section, we investigate the impact of different routing algorithms on the performance of the distributed cache architecture. As mentioned in Chapter 3, we experiment with three routing algorithms: widest-shortest, shortest-widest [Gue97-2], and a competitive routing algorithm [Hao2000]. Results generally show that the distributed cache architecture performs similarly under the different routing algorithms. This is especially true at higher arrival rates. Figure 4.9-a presents the impact of the routing algorithms on the cache utilization and the cache hit ratio, respectively. Figure 4.9-b presents the interaction between different routing algorithms and the reduction in the route computing load. We summarise our results as follows.

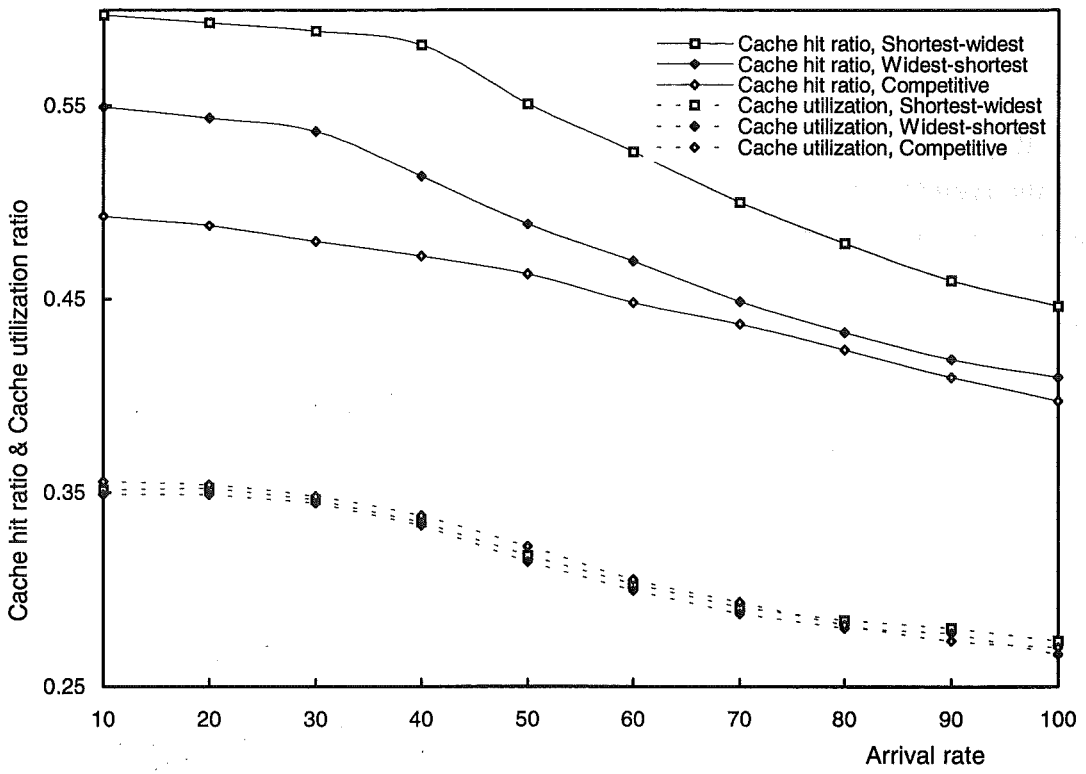
1. There is no significant relationship between the adopted routing algorithm and the cache utilization ratio. This is because the cache utilization ratio is mainly affected by the number and variety of the cached routes.

2. There seems to be a relationship between the adopted routing algorithm and the cache hit ratio. The shortest-widest routing algorithm achieves the best hit ratio, especially at lower arrival rates. It attempts to minimize the resource consumption by selecting the routes with a minimum hop count.

3. As presented in Figure 4.9-b, it seems that the choice of routing algorithm has only a marginal effect on the reduction of the route computing load. One also observes in Table 4.4 that all routing algorithms achieve similar results in terms of the reduction in the route computing load. This indicates that the distributed cache architecture maintains almost the same performance under different routing algorithms. This is an advantage because the distributed cache architecture can be incorporated in different networking technologies that may use different routing algorithms.

Routing Algorithm	***10 calls/sec***	***50 calls/sec***	***100 call/sec***
Shortest-Widest	20.9%	17.4%	11.2%
Widest-Shortest	19.1%	15.3%	10.9%
Competitive	17.5%	14.9%	10.7%

Table 4.4: Impact of routing algorithms on the reduction in route computing load



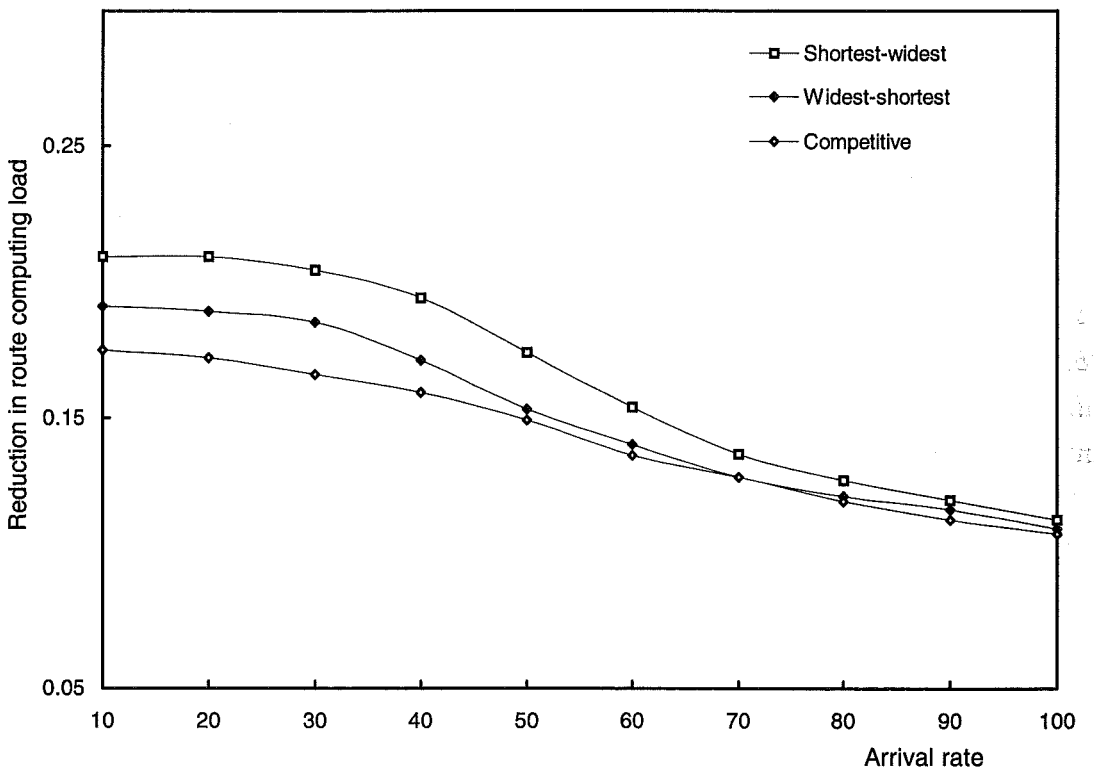


Figure 4.9-a: Impact of routing algorithms
Figure 4.9-b: Reduction of the route computing load
Network(10,type-2,full-mesh), 80% voice, MRC route selection policy
LRC flushing policy, flushing size = 2

4.3.5 Conclusions about the Route Selection Policies

In this section, we investigated the interaction between the route selection policies and the reduction in the route computing load in the distributed cache architecture, under a variety of different network and traffic conditions.

As a general observation, the choice of the route selection policy affects the on-demand route computing load. The route selection policy makes a non-trivial difference in the cache hit ratio, but does not seriously influence the cache utilization ratio. In all cases, the MRC route selection policy consistently outperforms other route selection policies, although the difference between the route computing load achieved under the different route selection policies can vary significantly, depending on the arrival rate, the traffic mixture, and the network size. We observe that the size of the inter-domain network can affect both the cache utilization ratio and the cache hit ratio.

The choice of topology aggregation affects the cache utilization ratio, while it does not seriously influence the cache hit ratio. The star aggregation technique performs poorly in terms of a reduction in the route computing load. This is true especially at higher arrival rates.

The impact of the traffic mixture can be quite significant, especially at higher arrival rates, because it affects both the cache utilization ratio and the cache hit ratio. In general, under video-dominated traffic, the distributed cache architecture is less effective in reducing the on-demand route computing load. However, the advantage of the MRC policy is more obvious when the network traffic is video-dominated.

Quite interestingly, the choice of routing algorithm has only a minimal impact on the performance of the distributed cache architecture in terms of a reduction in the route computing load. This indicates that the distributed cache architecture can be incorporated into different networking technologies and standards that may use different routing algorithms.

4.4 Cache Flushing

In Chapter 2, cache flushing was introduced as the cache content management technique designed for the distributed cache architecture. Cache flushing works as a background process and always maintains some free space in the caches across the border nodes. Therefore, the distributed cache architecture can operate smoothly, and operations such as route caching do not have a problem with the availability of sufficient space in the caches across the border nodes. This is accomplished by initiating a flushing procedure each time a cache at a border node becomes full. Flushing policies and flushing size are two influential factors that are studied in this section.

4.4.1 Route Computing Load vs. Cache Flushing

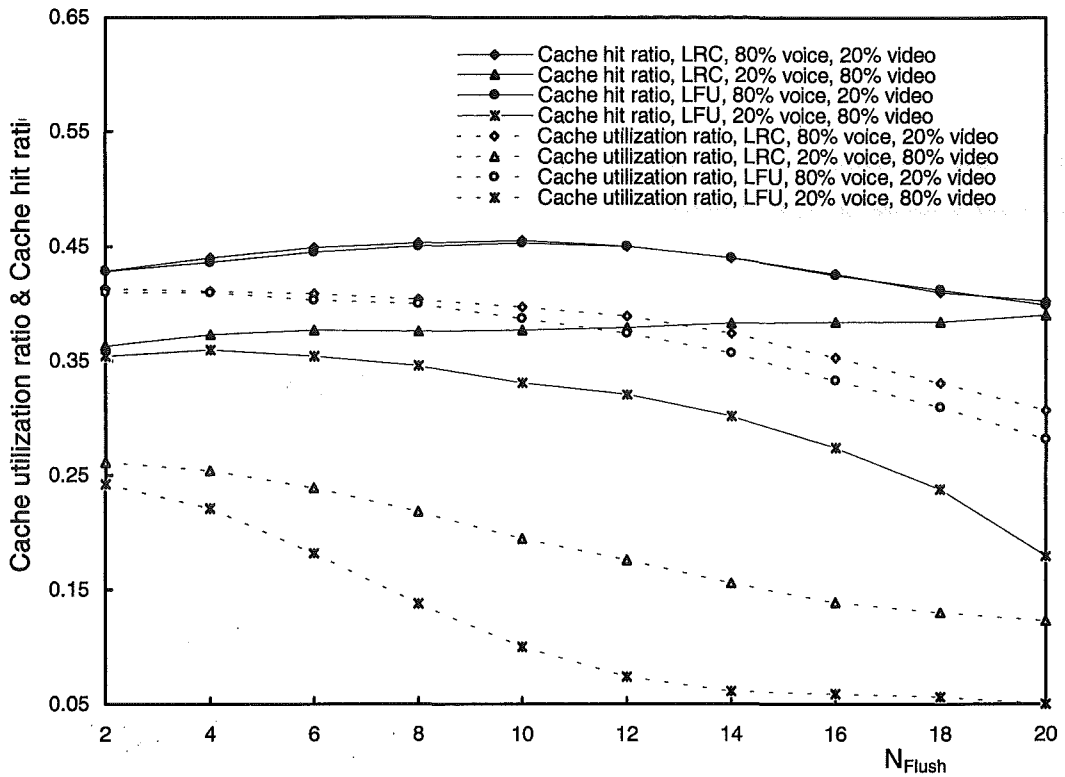
In Figures 4.10-a and 4.10-b, we investigate the relationships between several important aspects of cache flushing.

1. As a general observation, the flushing policies and flushing size affect both the cache utilization ratio and the cache hit ratio. The cache utilization ratio is affected because during flushing at a border node, useful routes can be mistakenly flushed out (i.e., erased) from the caches, especially when the flushing size is very large. The cache hit ratio is affected, because, based on the adopted flushing policy and the flushing size, the flushing may fail to identify those routes that are most likely to be obsolete. In addition, the propagation delay in the network links causes temporary inconsistencies between different segments of an end-to-end route that are stored in different caches at the border nodes. Therefore, a new call at a border node may select an *"apparently"* feasible route that has already been actually flushed out somewhere in the network, but the corresponding flushing message has not yet been received by the border node. This is more likely to happen in a larger network with longer links.

2. Under voice-dominated traffic, we see that both LRC and LFU flushing policies lead to similar cache hit ratios. In addition, under voice-dominated traffic, the hit ratio is not very sensitive to the flushing size. This is because a large number of routes are cached and the blocking ratio is relatively low so that the caches at the border nodes are quickly filled even with a large flushing size.

3. Under voice-dominated traffic, by increasing the flushing size from 2 to 10, the cache hit ratio slightly increases. The reason is that, initially, increasing the flushing size leads to the selection of more routes that are likely to be obsolete. However, under voice-dominated traffic, the cache hit ratio is reasonably stable.

4. Under video-dominated traffic, the choice of flushing policy can greatly affect the cache hit ratio because of the higher network state fluctuations. In this case, especially coupled with a large flushing size, the LRC outperforms the LFU flushing policy.



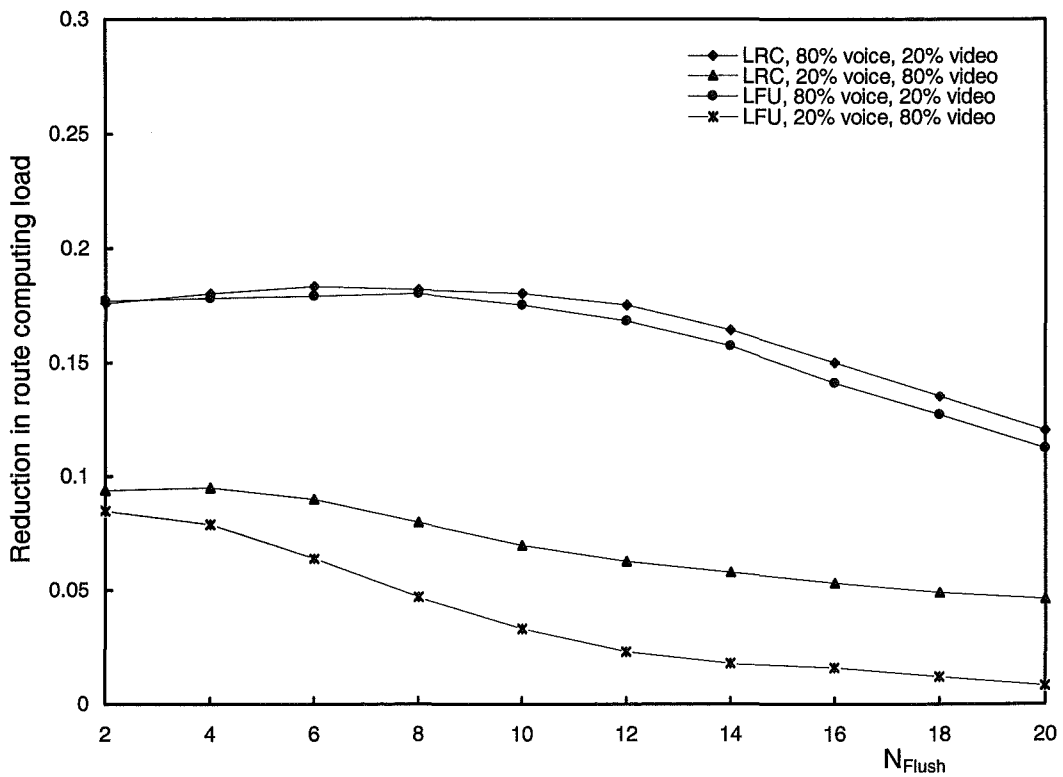


Figure 4.10-a: Impact of flushing policies and flushing size on the cache hit ratio and the cache utilization ratio

Figure 4.10-b: Route computing load vs. flushing size under different flushing policies

MRC route selection policy, Network(19,type-2,full-mesh)

5. In terms of the cache utilization ratio, the LRC policy consistently outperforms the LFU policy. However, we observe that the cache utilization ratio is very sensitive to the traffic mixture, especially when the flushing size is large.

6. In Figure 4.10-b, we observe that under both voice and video-dominated traffic, increasing the flushing size adversely affects the achieved reduction in the route computing load. This is true for both the LRC and LFU flushing policies. Note that when a small flushing size is used, the choice of policy does not make any difference to the reduction in the route computing load. (see Table 4.5)

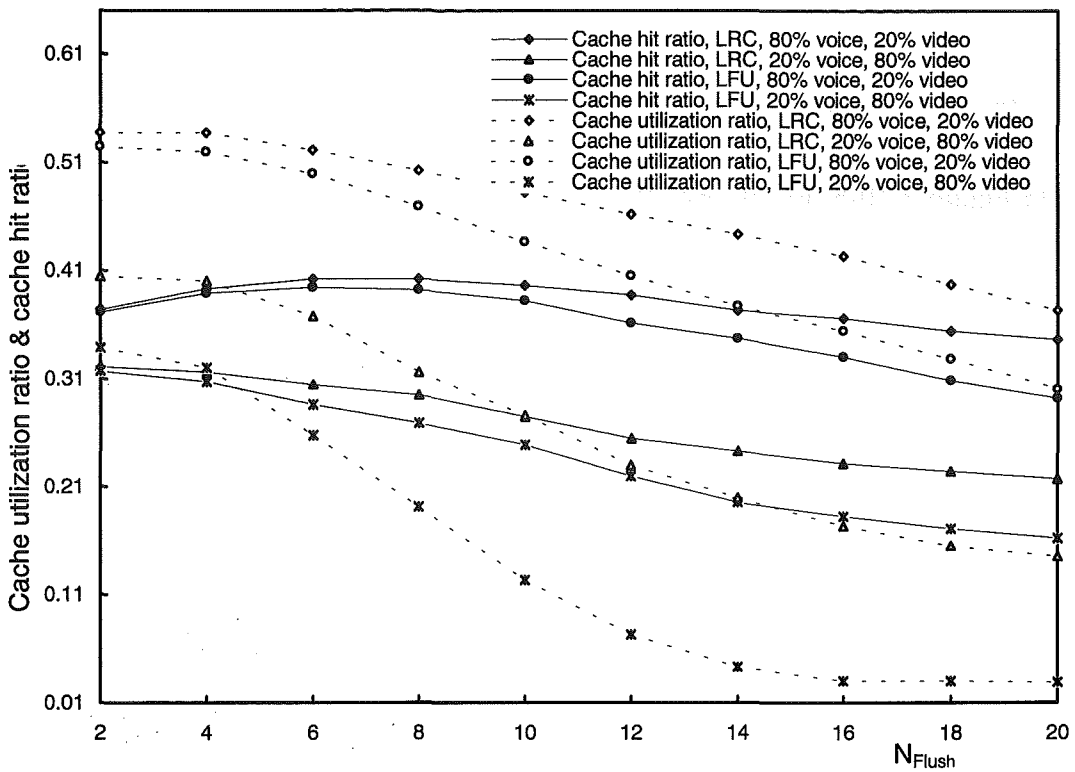
Cache Flushing policy	$N_{Flush} = 2$	$N_{flush} = 10$	$N_{flush} = 20$
LRC (voice-dominated traffic)	17.7%	18%	12%
LFU (voice-dominated traffic)	17.7%	17.5%	11.2%
LRC (video-dominated traffic)	9.4%	7%	4.6%
LFU (video-dominated traffic)	8.5%	3.3%	0.8%

Table 4.5: Cache flushing policies vs. reduction in the route computing load (Network with 19 domains)

4.4.2 Impact of Network Size

The experiment discussed in the previous section is repeated with a smaller network size of ten domains to observe the impact of the network size. We see that the cache utilization ratio is significantly affected by the choice of flushing policy. In comparison to the larger network, the cache utilization ratio seems to be more sensitive to the flushing size, particularly under video-dominated traffic. For example, in the previous experiment, where the network has nineteen domains, the cache utilization ratio reduces from 0.26 to 0.15 (i.e., a 38% drop) by increasing the flushing size from 2 to 20. Here, with a network size of 10 domains, the cache utilization ratio reduces from 0.4 to 0.14 (i.e., a 65% drop). The reason for this difference is that the smaller network has less capacity, so that fewer calls can be accepted and stored in the cache.

Finally, as Figure 4.11-b shows, greater reduction in the route computing load is achieved in a smaller network. However, the difference is not significant. This corresponds to a higher cache utilization ratio in a smaller network.



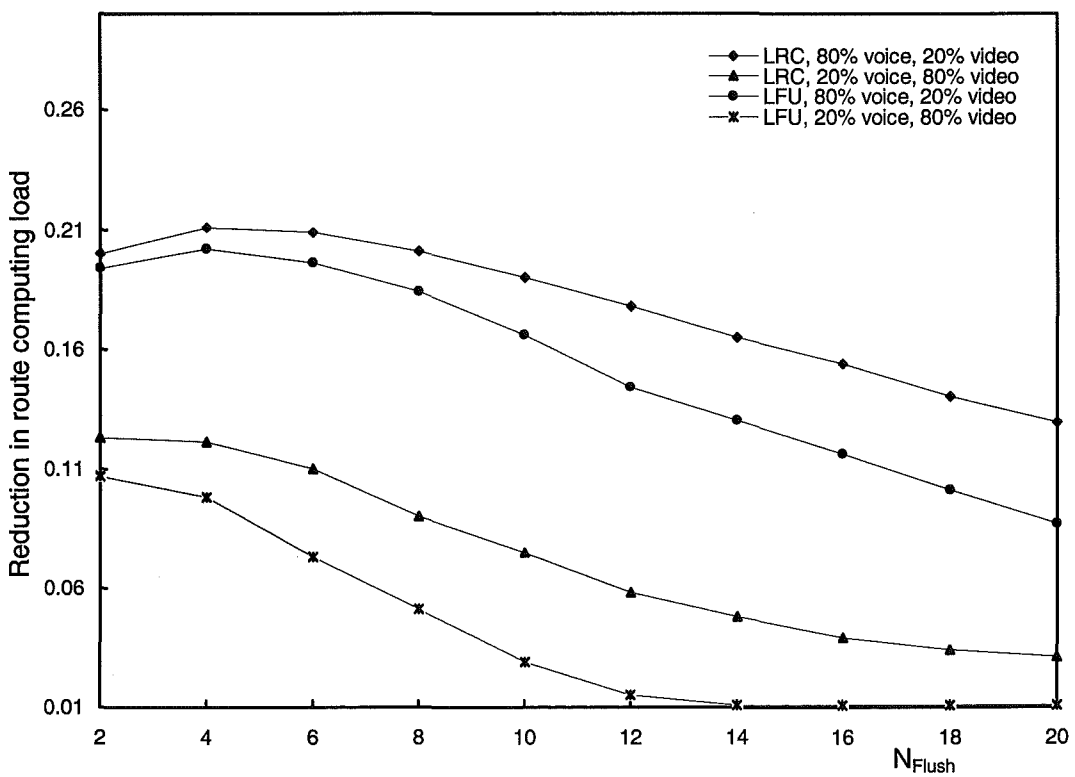


Figure 4.11-a: Impact of flushing policies and flushing size on the cache hit ratio and the cache utilization ratio

Figure 4.11-b: Route computing load vs. flushing size under different flushing policies

MRC route selection policy, Network(10,type-2,full-mesh)

Table 4.6 also presents a summary of the results displayed in Figure 4.11-b. One can see that under a small flushing size, the route computing load is reduced more effectively.

Cache Flushing policy	$N_{Flush} = 2$	$N_{flush} = 10$	$N_{flush} = 20$
LRC (voice-dominated traffic)	20%	19%	12.8%
LFU (voice-dominated traffic)	19.4%	16.6%	8.4%
LRC (video-dominated traffic)	12.3%	7.5%	0.3%
LFU (video-dominated traffic)	10.7%	2.9%	0.01%

Table 4.6: Cache flushing policies vs. reduction in the route computing load (Network with 10 domains)

4.4.3 Impact of Cache Flushing on the Bandwidth Blocking Ratio

Our investigations have shown that a smaller flushing size is preferred to a larger one. In this section, we investigate the impact of the flushing size and flushing policies on the network bandwidth blocking ratio. The overall bandwidth blocking ratio is important as it can adversely affect the overall performance of the routing.

In Figure 4.12, we have presented two sets of results obtained by applying the star and the full-mesh topology aggregation, respectively. The reason for considering the different topology aggregation techniques in this experiment is their significant influence on the bandwidth blocking ratio.

The overall bandwidth blocking ratio under star aggregation is much higher than that achieved under full-mesh aggregation. This is expected because in comparison to full-mesh aggregation, the star aggregation causes less accurate network states. Under all conditions, the bandwidth blocking ratio increases by increasing the flushing size. To this end, we can strongly conclude that a small flushing size should be used for cache flushing to avoid an excessive bandwidth blocking ratio. Finally, we observe that the LRC flushing policy is quite sensitive to the network state update interval. This is because a longer network state update interval reduces the accuracy of the network state information so that fewer routes can be successfully computed and cached.

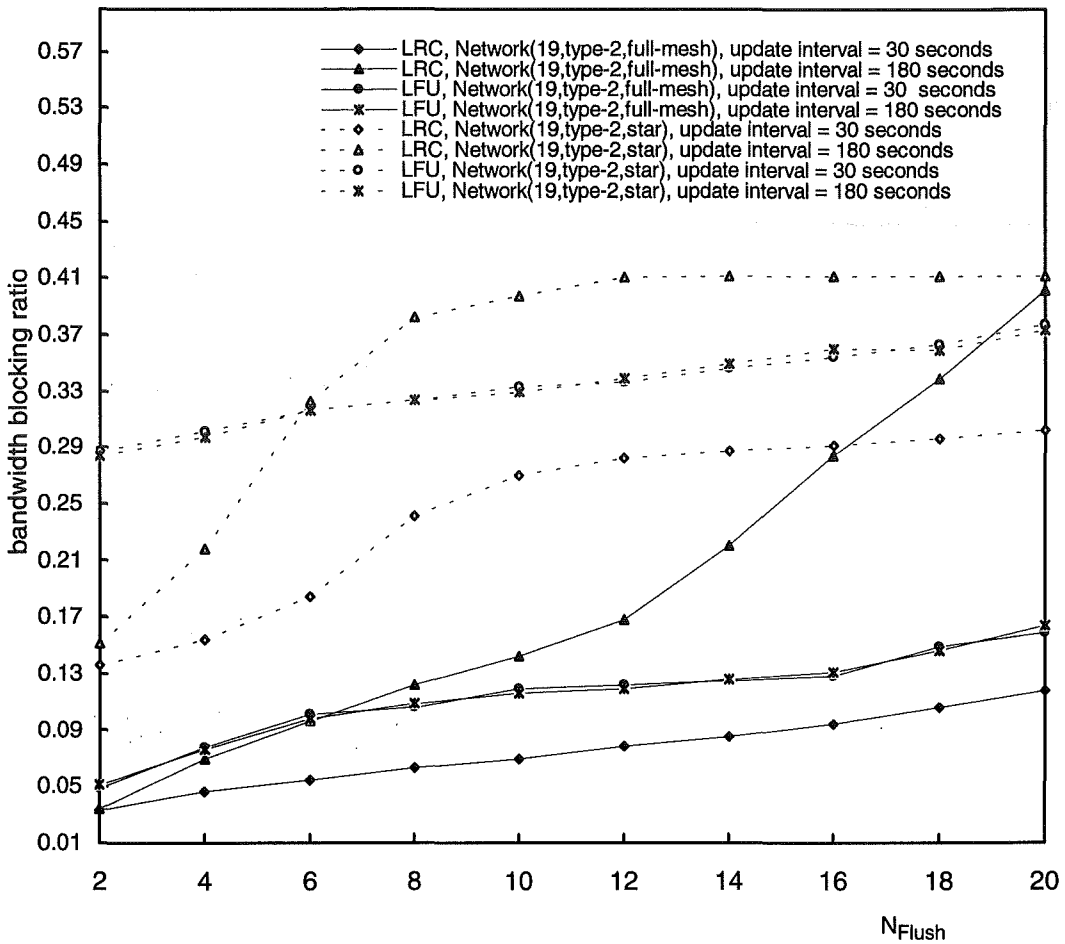


Figure 4.12: Impact of flushing policies and the flushing size on the bandwidth blocking ratio
MRC route selection policy

4.4.4 Conclusions about the Cache Flushing

We have found several key results related to cache. Firstly, cache flushing affects both the cache hit ratio and the cache utilization ratio. This is true especially under video-dominated traffic. Therefore, cache flushing should be carefully tuned to effectively contribute to reducing the route computing load, otherwise it can adversely affect the performance of the distributed cache architecture. Secondly, it is quite clear that a small flushing size is preferred; the bandwidth blocking ratio increases with an increase flushing size. In addition, with a smaller flushing size, the reduction of the route computing load is more significant. In the remainder of this thesis, we use a flushing size of two in all experiments. Thirdly, the LRC flushing policy consistently performs better than the LFU. Although the difference is trivial with a small flushing size, in the remainder of this thesis we conduct all our experiments with LRC.

4.5 Related Work

The works in [Pay97-1, Apo98-1] proposed a centralized cache architecture, assuming very simple and unrealistic network architectures. However, they did show that route caching is an effective technique in reducing the route computing load. Although, our approach is fundamentally different, our studies in this chapter confirm that route caching is quite effective in reducing the route computing load.

The proposals in [Pay97-1, Apo98-1] consider an on-demand route replacement mechanism, where a cached route is replaced when the cache at a node is full. In contrast, we have adopted a distributed cache content management system called cache flushing. It is worth mentioning that the cache flushing was not designed for a performance advantage over on-demand replacement mechanisms, rather it was designed because of the distributed nature of the proposed cache architecture.

We have also shown the influential role of the network architecture and topology aggregation on the performance of the distributed cache architecture.

4.6 Summary

In this chapter, we conducted thorough and studies on important aspects of the core distributed cache architecture under a variety of network and traffic conditions. The general goal was to demonstrate the effectiveness of the distributed cache architecture in reducing the on-demand route computing load. The cache utilization ratio and the cache hit ratio have also been studied, as well as the overall reduction in the route computing load. The cache utilization ratio indicates *how well the distributed cache architecture is used* by arriving calls, while the cache hit ratio measures *the accuracy of the cached routes*. Our findings are as follows:

1. The size of the cache elements significantly influences the cache utilization ratio. We have attempted to identify a proper size for the cache elements that maximizes the cache utilization ratio, while minimizes the memory consumption at the border nodes. Depending on the size and topology of the network, the aggregation technique, and the traffic conditions, allocating 70 to 100 entries for each of the cache elements maximizes the cache utilization ratio. We assume 100 entries for each cache element.

2. The route selection policy significantly influences the cache hit ratio and consequently the reduction in the route computing load. The MRC selection policy consistently outperforms other policies for different network sizes. Table 4.7-a summarises the best results achieved under the MRC selection policy for

two different networks and for voice-dominated traffic. The reduction in the route computing load is expressed in a percentage format. Table 4.7-b shows similar results for video-dominated traffic.

Route Selection Policy	***10 calls/sec***	***50 calls/sec***	***100 call/sec***
MRC, Network with 10 domains (voice-dominated traffic)	26.6%	20.12%	13.8%
MRC, Network with 19 domains (voice-dominated traffic)	18%	13.5%	10.3%
MRC, Network with 10 domains (video-dominated traffic)	18%	12%	6.1%

Table 4.7: The reduction in the route computing load under the MRC route selection policy

The choice of topology aggregation affects the cache utilization ratio, but has little or no impact on the cache hit ratio. Full-mesh aggregation leads to a higher cache utilization ratio. Table 4.8 presents the reduction in the route computing load achieved under each aggregation technique.

Route Selection Policy	***10 calls/sec***	***50 calls/sec***	***100 call/sec***
Full-mesh aggregation	26.5%	20.3%	14%
Star aggregation	23.3%	14.5%	9.3%

Table 4.8: The interaction between the topology aggregation and the reduction in the route computing load.
(Network with 10 domains, voice-dominated traffic)

3. The choice of routing algorithm has only a minimal impact on the performance of the distributed cache architecture in terms of a reduction in the route computing load. This indicates that the distributed cache architecture can be incorporated into different networking technologies and standards that may use different routing algorithms.

4. Cache flushing affects both the cache hit ratio and the cache utilization ratio, especially under video-dominated traffic. Therefore, cache flushing should be carefully tuned to effectively contribute to reducing the route computing load. The studies suggest that a smaller flushing size is preferred to a larger one. It is also shown that the LRC flushing policy consistently outperforms the LFU.

Although the results indicate that the core distributed cache architecture is quite effective in reducing the route computing load, we argue that by improving the cache hit ratio and the cache utilization ratio, the performance of the core distributed cache architecture can be significantly improved.

Chapter 5

Performance Evaluation of Cache Snooping

5.1 Introduction

The simulation studies conducted on the core distributed cache architecture in Chapter 4 indicate that it is quite effective in reducing the on-demand route computing load. However, the percentage of the reduction in the route computing load varies significantly under different network and traffic conditions. As concluded in Chapter 4, as far as the reduction of the route computing load is concerned, the performance of the distributed cache architecture can be improved by increasing the cache utilization and the cache hit ratio. In this chapter, we focus on *improving the cache hit ratio* using cache snooping.

As detailed in Chapter 2, changes in the network states can cause cached routes to become obsolete (i.e., unable to offer their associated bottleneck bandwidth). While this does not influence the cache utilization ratio, it can seriously decrease the cache hit ratio. Our investigations in Cchapter 4 have confirmed this problem, especially when the network state changes rapidly (e.g., at high arrival rates and/or when the network traffic is video-dominated). In addition, because the distributed cache architecture stores routes in the form of interconnected segments, an end-to-end route becomes obsolete when even one of its segments is obsolete. This causes greater on-demand route computing load by reducing the likelihood of an end-to-end route being reusable. Cache snooping is proposed as a distributed technique to alleviate the impact of the network state fluctuations on the accuracy of the cached routes so that a higher cache hit ratio can be maintained. In this chapter we investigate different aspects of cache snooping in an attempt to identify the important trade-offs between

the reduction in the route computing load, and the cache snooping parameters and policies.

5.2 Cache Snooping vs. Route Computing Load

The goal of cache snooping is to increase the cache hit ratio so that the overall route computing load is reduced more effectively. In this section, we evaluate the general impact of the cache snooping on the cache hit ratio and the route computing load, under voice and video-dominated traffic respectively. We use two different network sizes and compare the results with those obtained without using cache snooping.

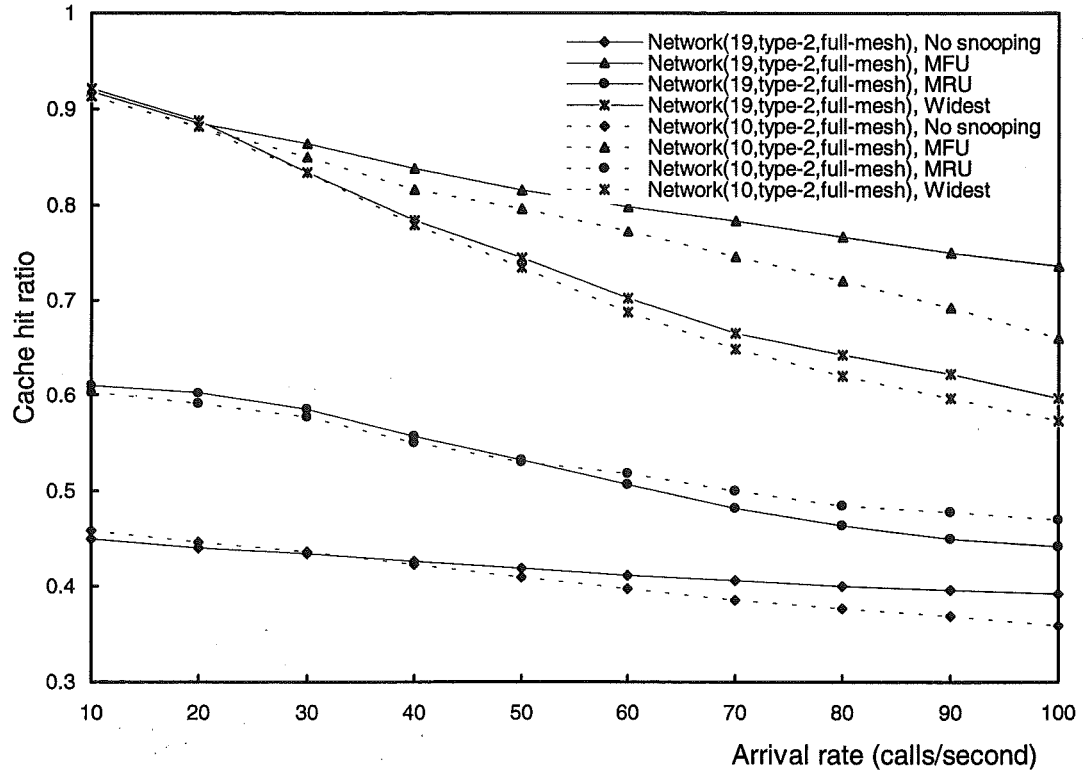
5.2.1 Voice-dominated Traffic

In Figure 5.1-a, we study the impact of cache snooping on the cache hit ratio. The main observations are as follows:

1. As an immediate observation, cache snooping, even at higher arrival rates, significantly increases the cache hit ratio under all conditions, although it is greatly influenced by the snooping policy.

2. Assuming the same network size, the MFU policy leads to the highest cache hit ratio. This is probably because MFU attempts to increase the accuracy of those routes that are most likely to be reused (i.e., they have been used most frequently in the past). We observe that in a network with nineteen domains, without applying the cache snooping, the cache hit ratio varies between 45% and 39%, depending on the call arrival rate. At the same time, under cache snooping with MFU as the snooping policy, the cache hit ratio varies between 92% and 74%, depending on the call arrival rate. Comparatively speaking, in this case the cache hit ratio has been increased between 104% and 89%.

3. At lower arrival rates, the size of the inter-domain network makes little or no difference, while at higher arrival rates, a smaller network achieves a lower cache hit ratio. This is because of the higher network state fluctuations in a smaller network. For example, under the MFU snooping policy and an arrival rate of 100 calls/second, the cache hit ratio achieved in a network with ten domains is 8% less than that achieved in a network with nineteen domains.



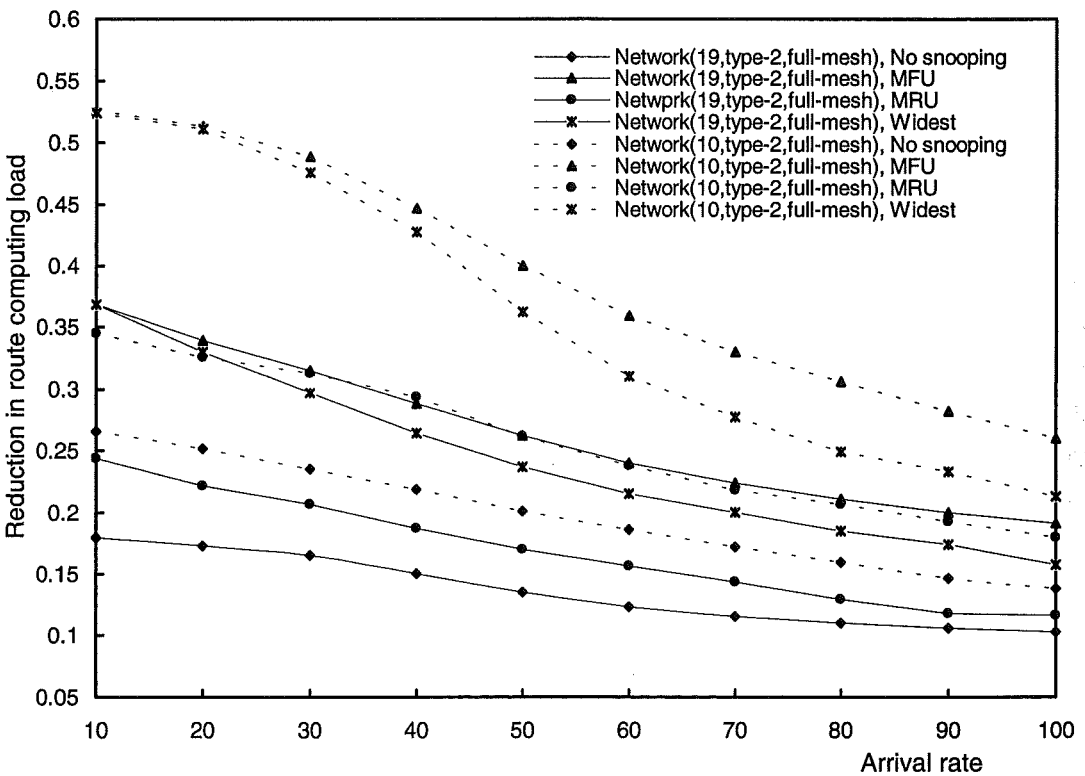


Figure 5.1-a: Impact of snooping policies on the cache hit ratio
Figure 5.1-b: Impact of the cache snooping on the route computing load
Snooping interval = 30 seconds, Snooping size = 6
80% voice, MRC route selection policy
LRC flushing policy, flushing size = 2

4. Although cache snooping significantly increases the cache hit ratio, it does not lead to an overall stable cache hit ratio, when the arrival rate is increased. This means that cache snooping loses its effectiveness to some degree when the network is heavily loaded.

5. Figure 5.1-b demonstrates the effectiveness of cache snooping in significantly reducing the on-demand route computing load. Table 5.1-a and Table 5.1-b present simplified sets of results corresponding to the graphs presented in Figure 5.1-b. For example, for a network with ten domains, under the core distributed cache architecture (i.e., without snooping), the route computing load is reduced between 26.6% to 13.8%, depending on the call arrival rate. In the same network, cache snooping coupled with the MFU snooping policy, reduces the route computing load by 52.4% to 25.12%, depending on the call arrival rate. We observe similar results obtained for a network with nineteen domains.

6. Similarly to the case for the cache hit ratio, we see that cache snooping becomes less effective in reducing the route computing load when the arrival rate is increased. For example, as Table 5.1-a shows, assuming a network with ten domains and the MFU policy, increasing the arrival rate from 10 to 100 calls/second causes the reduction in the route computing load to drop from 52.4% to 25.12%.

Snooping Policy	***10 calls/second***	***50 calls/second***	***100 call/second***
MFU	52.4%	38.15%	25.12%
Widest	52.4%	36%	21%
MRU	34%	26.15%	18%
No Snooping	26.6%	20.1%	13.8%

(a)

Snooping Policy	***10 calls/second***	***50 calls/second***	***100 call/second***
MFU	36.8%	27.15%	19.1%
Widest	36.8%	23.7%	15.7%
MRU	24.4%	17%	11.6%
No Snooping	18%	12.5%	9.8%

(b)

Table 5.1: Impact of snooping policies on the reduction in the route computing load under voice-dominated traffic
(a): Network with 10 domains, (b): Network with 19 domains

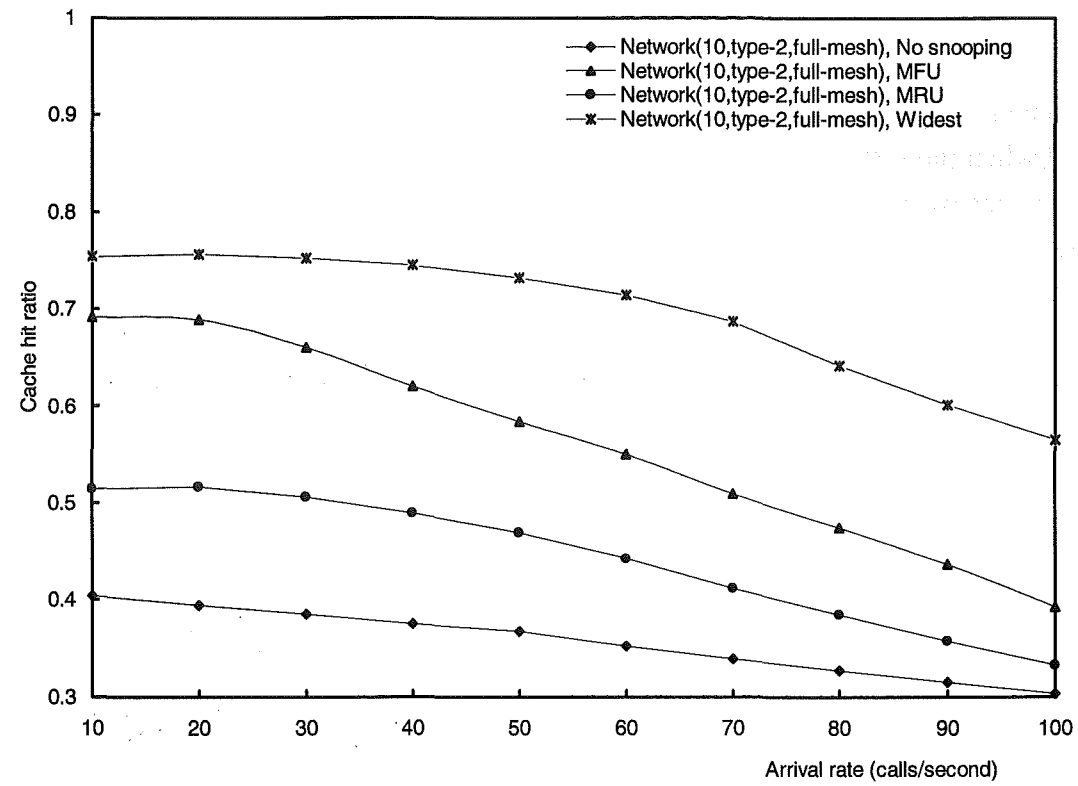
5.2.2 Video-dominated Traffic

We investigate the effectiveness of cache snooping in reducing the route computing load when the network traffic is video-dominated. This is important because a video-dominated traffic mixture leads to a highly variable network state so that the cache hit ratio is decreased. Figure 5.2-a shows the impact of cache snooping on cache hit ratio. Similar to a network with voice-dominated traffic, the cache snooping leads to a much higher cache hit ratio. The results of our experiments in the previous section suggested that under voice-dominated traffic, the MFU snooping policy consistently outperformed the other snooping policies. In contrast, we observe that under video-dominated traffic, the widest snooping policy outperforms the MFU. We also notice that the difference between the widest snooping policy and the other policies becomes larger by increasing the call arrival rate. The reason is that under a video-dominated traffic, the majority of the cached routes are wide. Under the widest snooping policy, by increasing the mean arrival rate from 10 to 100 calls/second, the cache hit ratio decreases by 20% (from 75% to 55%). At the same time, under the core cache architecture (i.e., no snooping) the cache hit ratio decreases by 9% (from 40% to 31%).

Figure 5.2-b presents the reduction in the route computing load, achieved by different snooping policies under video-dominated traffic. The widest snooping policy is most effective in reducing the route computing load. Table 5.2 displays a set of numerical values for the reduction in the route computing load corresponding to the Figure 5.2-b. One observes that by increasing the arrival rate, cache snooping becomes less effective. For example, under the widest snooping policy, with an arrival rate of 10 calls/second, the route computing load is reduced by 34%. At the same time, assuming an arrival rate of 100 calls/second, only a 10.5% reduction in the route computing load is achieved. The reason for this phenomenon is that the network state at a higher arrival rate is highly variable, and cache snooping fails to completely capture the rapid changes in the network states.

Snooping Policy	***10 calls/second***	***50 calls/second***	***100 call/second***
Widest	34%	25%	10.5%
MFU	31.3%	19.4%	8%
MRU	23.2%	15.2%	7%
No Snooping	18.1%	12%	6.3%

Table 5.2: Impact of snooping policies on the reduction in the route computing load under video-dominated traffic, Network with 10 domains



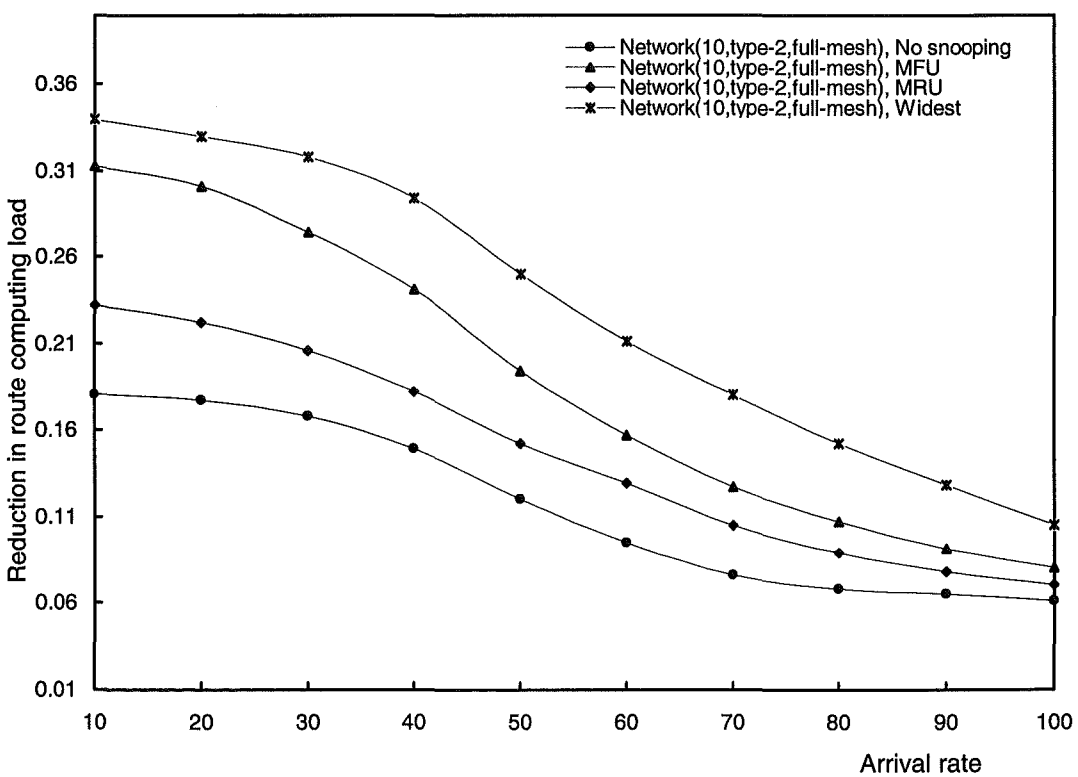


Figure 5.2-a: Impact of snooping policies on the cache hit ratio
Figure 5.2-b: Impact of snooping policies on the route computing load
Snooping interval = 30 seconds, Snooping size = 6
80% video, MRC route selection policy
LRC flushing policy, flushing size = 2

5.3 Impact of Snooping Interval under Different Snooping Policies

As detailed in Chapter 2, the snooping interval is one of the parameters associated with the cache snooping and defines how often cache snooping is initiated in the caches at the border nodes. We examine the impact of the snooping interval on the cache hit ratio. We present two sets of results obtained assuming low and high call arrival rate respectively. In both experiments we explore the effects of different snooping policies and different traffic mixtures.

5.3.1 Low Arrival Rate

In Figure 5.3, we assume the network arrival rate is 20 calls/second. The cache snooping interval varies between 30 and 300 seconds. In general, it appears that there is not a strong connection between the snooping policy and the snooping interval. The snooping interval seems to have similar effects on the cache hit ratio, regardless of the adopted snooping policy, although, as discussed in the previous section, the overall cache hit ratio achieved, is a function of the snooping policy. Our observations are as follows:

1. Generally, under all conditions, the cache hit ratio decreases by increasing the snooping interval. In other words, the cached routes are more likely to be obsolete under a longer snooping interval, as cache snooping keeps track of the changes in the network state less efficiently. In addition, we observe that the cache hit ratio is reduced reasonably smoothly (i.e., without any sudden change) even under video-dominated traffic, because the arrival rate is reasonably low (20 calls/second), so that few calls arrive at the border nodes during the snooping interval.

2. Under all conditions, as the snooping interval is increased, the cache hit ratio becomes less sensitive to any changes in the snooping interval. For example, increasing the snooping interval by more than 150 seconds causes very little change in the cache hit ratio, while increasing the snooping interval from 30 to 90 seconds makes a clear difference in the cache hit ratio. This is especially true under the video-dominated traffic. This is because under a video-dominated traffic, the changes in the network state occur more rapidly. A shorter snooping interval is can capture these rapid changes more effectively.

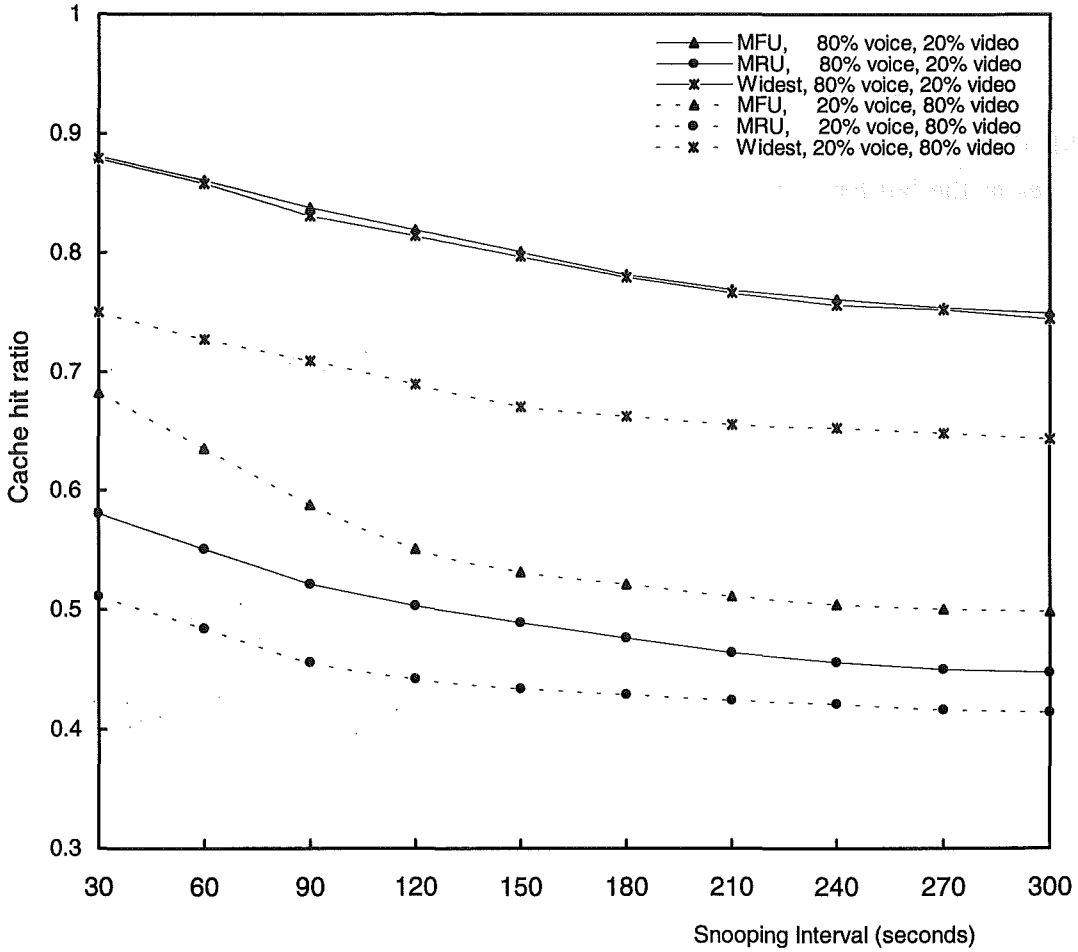


Figure 5.3: Impact of snooping size on the cache hit ratio under different snooping policies
Snooping size = 6
MRC route selection policy, LRC flushing policy, flushing size = 2

5.3.2 High Arrival Rate

The previous experiment is repeated with a much higher arrival rate. This increases the fluctuations in the network states so that we can investigate the impact of the snooping interval on the cache hit ratio when network states change rapidly. We assume an arrival rate of 80 calls/second. We observe several interesting phenomena as follows:

1. In comparison to the previous experiment that was conducted by a low call arrival rate, here, the cache hit ratio is much more sensitive to the snooping interval, even

under voice-dominated traffic. In comparison, assuming voice-dominated traffic, the MFU snooping policy, and 20 calls/second arrival rate, the cache hit ratio drops from 88% to 83% when the snooping interval is increased from 30 to 90 seconds (Figure 5.3). In contrast, assuming the same conditions, and 80 calls/second arrival rate, the cache hit ratio drops from 72% to 52% when the snooping interval is increased from 30 to 90 seconds (Figure 5.4). This increased sensitivity is because the higher arrival rate causes more fluctuations in the network states so that the cached routes become obsolete faster. Therefore, increasing the snooping interval causes a more rapid reduction in the accuracy of the cached routes. Under video-dominated traffic, the cache hit ratio is even more sensitive to the snooping interval, because network states change more rapidly due to the allocating and releasing of larger amounts of bandwidth across the network links for video calls.

2. At higher arrival rates, the MRU snooping policy behaves quite differently from other policies. The cache hit ratio under the MRU snooping policy is much less sensitive to the snooping interval. For example, under voice-dominated traffic, using the MRU snooping policy causes the cache hit ratio to drop from 47% to 43% when the snooping interval is increased from 30 to 90 seconds. With video-dominated traffic, using the MRU policy causes the cache hit ratio drops from 38% to 30%. The reason is that the MRU snooping policy chooses the cached routes that have been used most recently. These routes are the least likely to be obsolete, because an obsolete route is removed from the cache when the route setup procedure fails. Therefore, the choice of snooping interval is not very influential. For a similar reason, we observe that the general cache hit ratio under the MRU snooping policy is less than that obtained under other policies.

Our studies on the influence of the cache snooping interval suggests that a reasonably short snooping interval is desirable because it may maximize the cache hit ratio, which is translated into a higher reduction in the route computing load. In addition, it seems that those snooping policies that achieve a higher cache hit ratio are, in fact, more sensitive to the length of the snooping interval. This supports the choice of a reasonably short snooping interval. The network traffic mixture can greatly affect the performance of cache snooping, especially at higher arrival rates when the amount of available bandwidth on the network links changes rapidly. Additionally, the call arrival rate significantly affects the performance of cache snooping even under voice-dominated traffic.

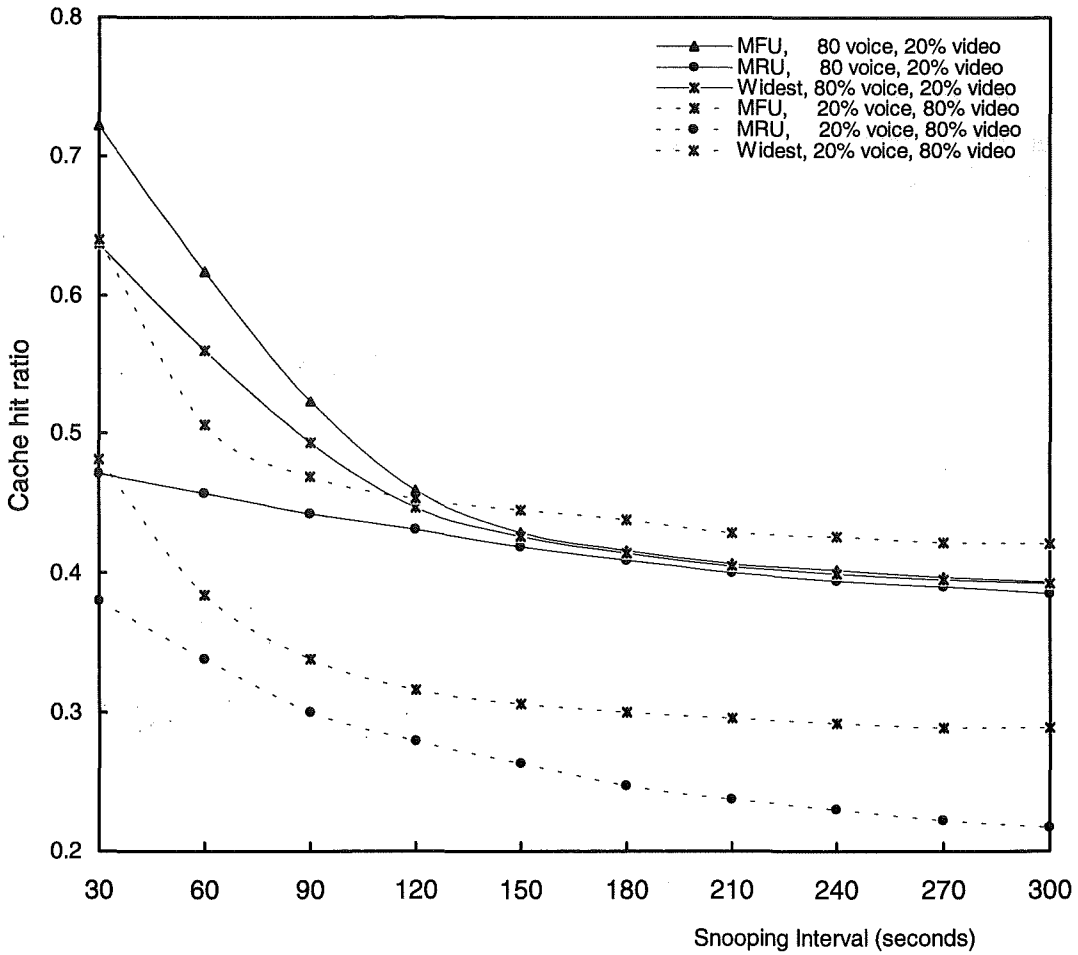


Figure 5.4: Impact of snooping interval on the cache hit ratio under different snooping policies, Snooping size = 6
MRC route selection policy, LRC flushing policy, flushing size = 2

5.4 Impact of Snooping Size under Different Snooping Policies

The snooping size (N_{snoop}) defines an upper bound on the number of segments that are snooped during each cache snooping at the caches of the border nodes. Note that cache snooping is performed only for segments that are in the *"available"* state. If the total number of *"available"* segments in a cache is more than or equal to N_{snoop}, then N_{snoop} segments will be selected. If the total number of *"available"* segments in a cache is less than N_{snoop}, then all of them will be selected. Here, we study the impact of the snooping size on the cache hit ratio under various conditions. We conduct separate experiments under voice and video-dominated traffic respectively. We also

attempt to explore the relationship between the snooping size, the snooping policies, and the call arrival rate.

5.4.1 Voice-dominated Traffic

Figure 5.5 shows the impact of the snooping size on the cache hit ratio, under different snooping policies and arrival rates. We have assumed voice-dominated traffic. The main observations are as follows:

1. Under all conditions and for all snooping policies, by increasing the snooping size, initially the cache hit ratio increases sharply up to a certain point, after which it increases only trivially. A larger snooping size means that more segments are selected for snooping. Initially, a very small snooping size (e.g., 2) can not effectively increase the accuracy of the caches at the border nodes because many segments that are likely to be used in the future are simply ignored during cache snooping. Therefore, with a very small snooping size, it is expected that the choice of snooping policy does not have a significant impact on the cache hit ratio. This is evident in Figure 5.5, where we observe that with a snooping size of 2, the difference between the cache hit ratio obtained under the different snooping policies is less than 8%. Increasing the snooping size to larger values causes the ignored segments to be included in the snooping procedure. This leads to a sharp increase in the cache hit ratio. In this situation, the choice of the snooping policy is expected to show a more significant difference in the cache hit ratio. This is evident in Figure 5.5, where by increasing the snooping size from 2 to 8, the difference between the cache hit ratio obtained under the different snooping policies is at least 20%. After this stage, increasing the snooping size has only a minimal effect on the cache hit ratio. This is because the segments are *sorted,* based on the adopted snooping policy, in a descending fashion so that a large snooping size selects many segments of little importance.

2. Quite interestingly, the results suggest that snooping policies that achieve a higher cache hit ratio show more sensitivity to the initial increase of the snooping size. For example, when the arrival rate is 20 calls/second, increasing the snooping size from 2 to 8 causes a 35% and 12% increase in the cache hit ratio under the MFU and MRU snooping policies respectively. The reason is that a more effective snooping policy uses better criteria to select segments for snooping, so that the initial increase in the snooping size causes a higher number of *better-handpicked* segments to be snooped.

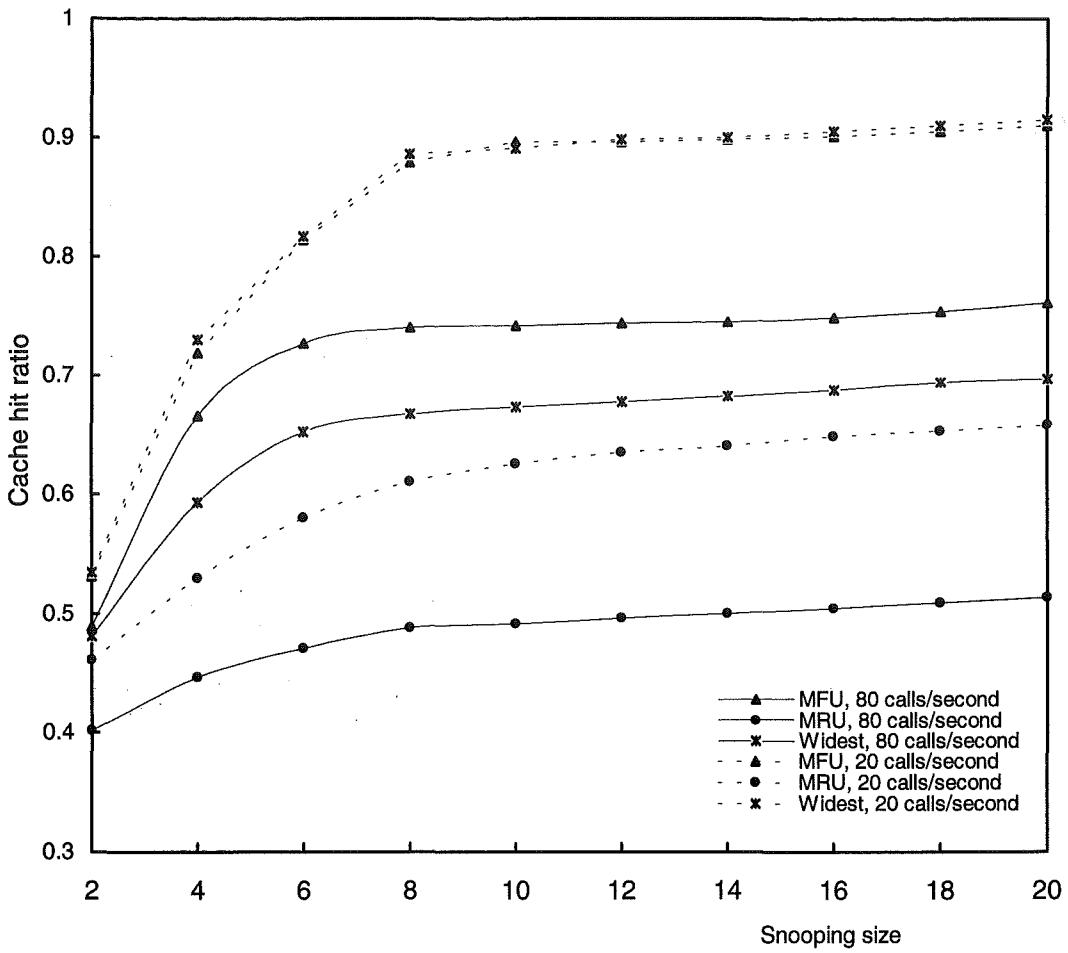


Figure 5.5: Impact of snooping size on the cache hit ratio under different snooping policies,
Snooping Interval = 30 seconds, 80% voice
MRC route selection policy, LRC flushing policy, flushing size = 2

5.4.2 Video-dominated Traffic

In Figure 5.6 we repeat the previous experiment with video-dominated traffic. Similarly, we observe that initially by increasing the snooping size, the cache hit ratio increases rapidly. However, under the video-dominated traffic, the cache hit ratio is more sensitive to the snooping size. For example, assuming an arrival rate of 80 calls/second and the MFU snooping policy, under voice-dominated traffic, by increasing the snooping size from 2 to 4, the cache hit ratio increases by 16%. In contrast, assuming the same conditions, under video-dominated traffic, by increasing the snooping size from 2 to 4, the cache hit ratio increases by 29%, because the video-

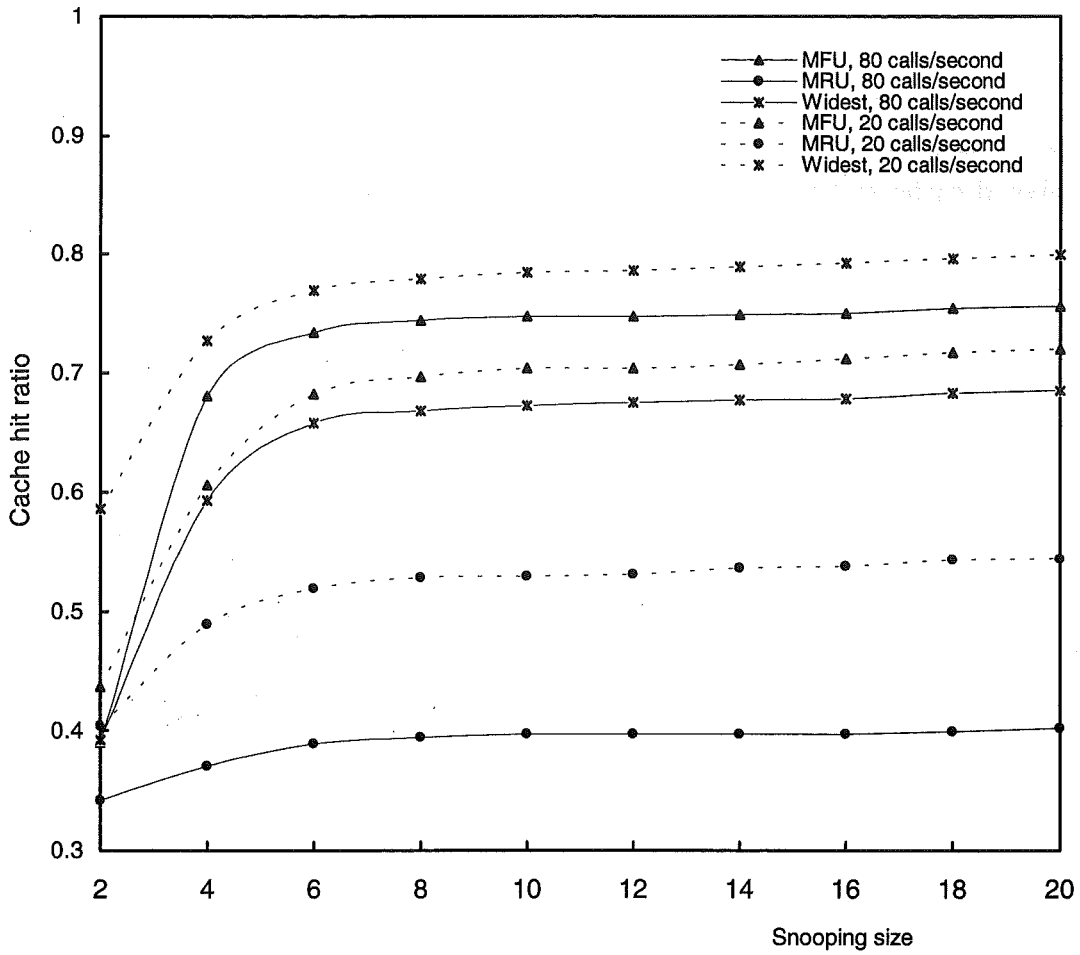


Figure 5.6: Impact of snooping size on the cache hit ratio under different snooping policies
Snooping Interval = 30 seconds, 80% video
MRC route selection policy, LRC flushing policy, flushing size = 2

dominated traffic causes larger and faster fluctuations in the network states. This adversely affects the accuracy of the cached routes. Similarly, we again observe that the MRU policy leads to the lowest cache hit ratio and is least sensitive to the snooping size. However, the difference between the MRU and other snooping policies is larger than that observed under for voice-dominated traffic. Finally, under video-dominated traffic a smaller snooping size is required to achieve a *reasonably stable* cache hit ratio. This is much fewer routes (i.e., fewer segments) are successfully computed and stored in the distributed cache architecture so that a smaller snooping size covers a larger percentage of the cached segments.

5.5 Impact of Inaccurate Network State Information

In Figure 5.7 we investigate effects of the cache snooping on the bandwidth blocking ratio, under video-dominated traffic. The important observation is that the distributed cache architecture, coupled with cache snooping is effective in reducing the bandwidth blocking ratio of the network, especially in the presence of highly inaccurate network state information (i.e., a long network state update interval). This is a secondary achievement for the distributed cache architecture. It is important because a lower blocking ratio means that more calls can simultaneously exist in the network. In other words, the overall performance of the routing is increased. In the absence of the distributed cache architecture, all calls are routed in an on-demand fashion based on the network state information stored at the LSDB of the border node. As outlined in Chapter 1, a long state update interval in the network, coupled with the topology aggregation, reduces the accuracy of the network state information so that the probability of a wrong routing decision is increased. This increases the overall blocking ratio of the network. On the other hand, in the presence of the distributed cache architecture, many of the arriving calls are routed by reusing the cached routes. In addition to reducing the on-demand route computing load, using the cached routes reduces the probability of making wrong routing decisions for two main reasons.

1. As explained in Chapter 2, the distributed cache architecture stores the detailed topology and network state information for each cached route in the form of several inter-connected segments so that the distributed cache architecture is completely de-coupled from the LSDB at the border nodes. In this way, once a feasible route is found in the cache, the corresponding route setup procedure can proceed without relying on the network state and the topology information stored in the LSDB of the border nodes. Therefore, the distributed cache architecture does not suffer from inaccurate network state information caused by long state update intervals. Furthermore, the novel segment-by-segment approach in storing an end-to-end route in the cache means that the cached routes do not suffer from the inaccuracy caused by topology aggregation.

2. As studied in this chapter, cache snooping alleviates the effects of the changes in the network states so that the accuracy of the cached routes is significantly increased. Cache snooping is performed in a distributed fashion by the caches at the border nodes using local segment information so that it does not rely on information stored in the LSDB of the border nodes.

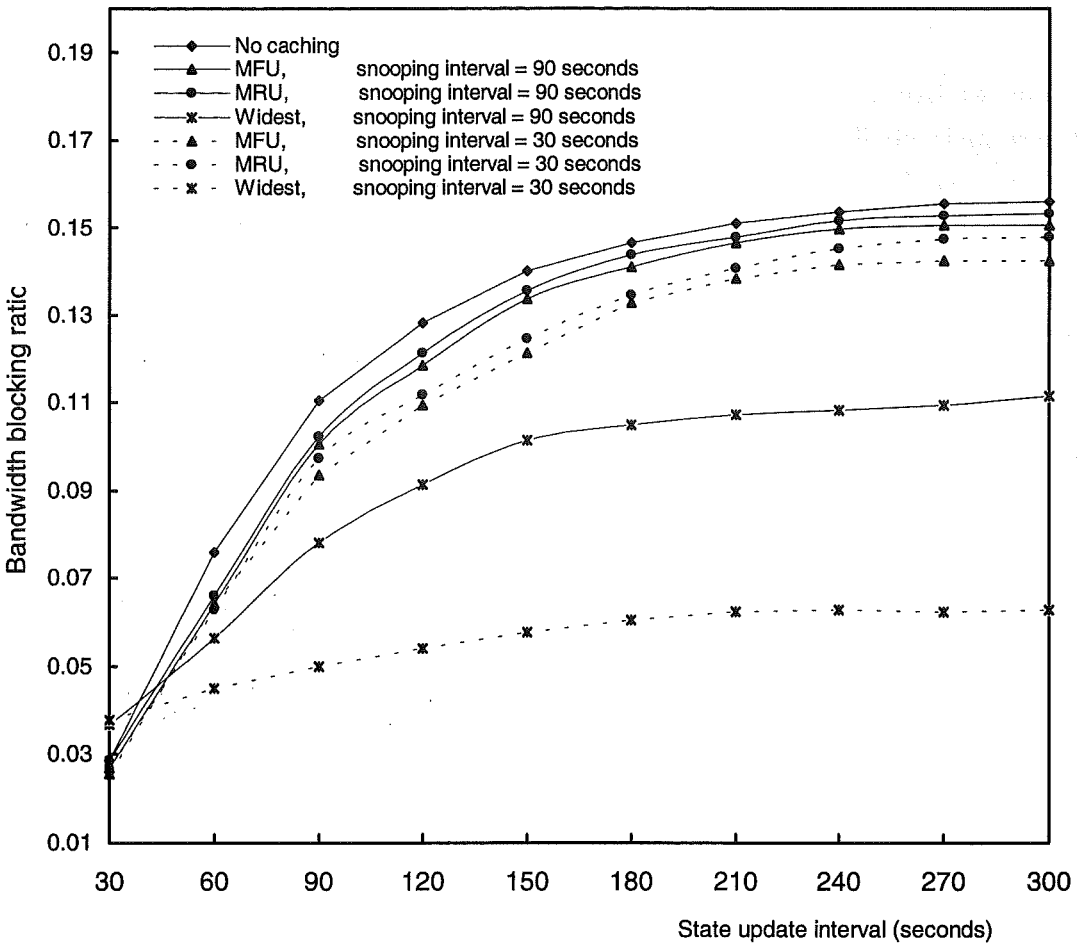


Figure 5.7: Impact of cache snooping on the bandwidth blocking ratio under different snooping policies and snooping intervals
MRC route selection policy, LRC flushing policy, flushing size = 2

We can observe in Figure 5.7 that using cache snooping with a reasonably short snooping interval leads to a lower bandwidth blocking ratio. For example, in the absence of the distributed cache architecture, a state update interval of 300 seconds leads to a bandwidth blocking ratio of about 16%. At the same time, under a cache snooping interval of 30 seconds, coupled with the widest snooping policy, a state update interval of 300 seconds leads to a bandwidth blocking ratio of about 6%. It is obvious that using the right snooping interval and snooping policy, cache snooping increases the tolerance of the routing in the presence of inaccurate network state information (i.e., a long state update interval).

5.6 An Alternative Approach: Active Snooping

So far we have assumed that when cache snooping detects an obsolete segment, the entire route is labelled as stale (i.e., removed from the cache). Here, we investigate the potential benefits of active snooping. As a brief reminder from Chapter 2, active snooping attempts to repair the damaged segment instead of removing it from the cache. This is accomplished by a segment re-computing procedure to compute a new segment to replace the old one. Segment re-computing is similar to the usual on-demand route computing procedure except that it computes a segment only between two adjacent border nodes. Because the border nodes have detailed topological information of their own network domain, the segment re-computing does not suffer from the inaccuracy caused by topology aggregation.

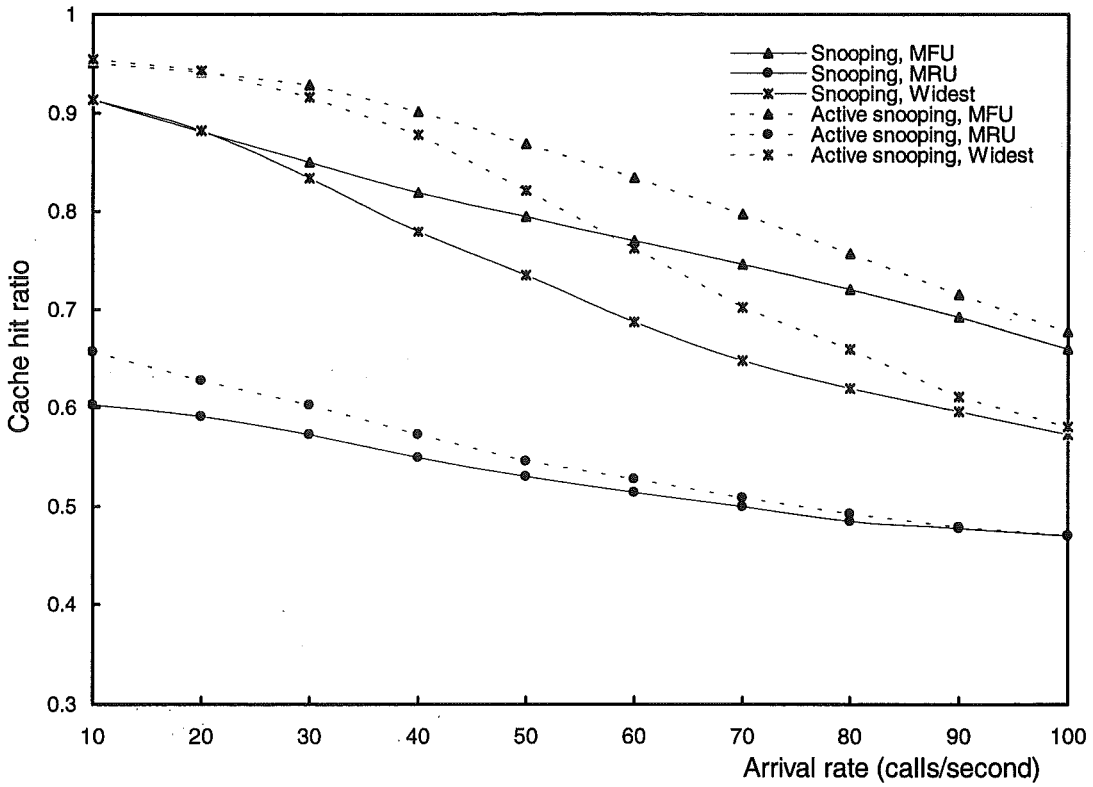
We investigate the impact of the active snooping on the cache hit ratio. In addition, we have a look at how active snooping can help to increase the tolerance of the routing to the inaccurate network state information. As we shall see, the active snooping shows little benefit in reducing the on-demand route computing load as it fails to efficiently increase the cache hit ratio. Instead, the main advantage of active snooping is increasing the tolerance of the routing, especially in the presence of highly inaccurate network state information caused by long network state update intervals.

5.6.1 The Impact on the Cache Hit ratio

Figure 5.8-a and Figure 5.8-b show the impact of active snooping on the cache hit ratio, under voice-dominated and video-dominated traffic respectively. We make several observations as follows:

1. Assuming voice-dominated traffic, regardless of the snooping policy, active snooping increases the cache hit ratio at lower arrival rates. However, the increase is not significant. At higher arrival rates, active snooping has little or no impact on the cache hit ratio, as the segment re-computing procedure is more likely to fail as network links have less available bandwidth.

2. Under video-dominated traffic, even at lower arrival rates, active snooping makes little or no difference to the cache hit ratio. This is because under video-dominated traffic the network links are heavily loaded and the likelihood of successfully re-computing a segment is decreased.



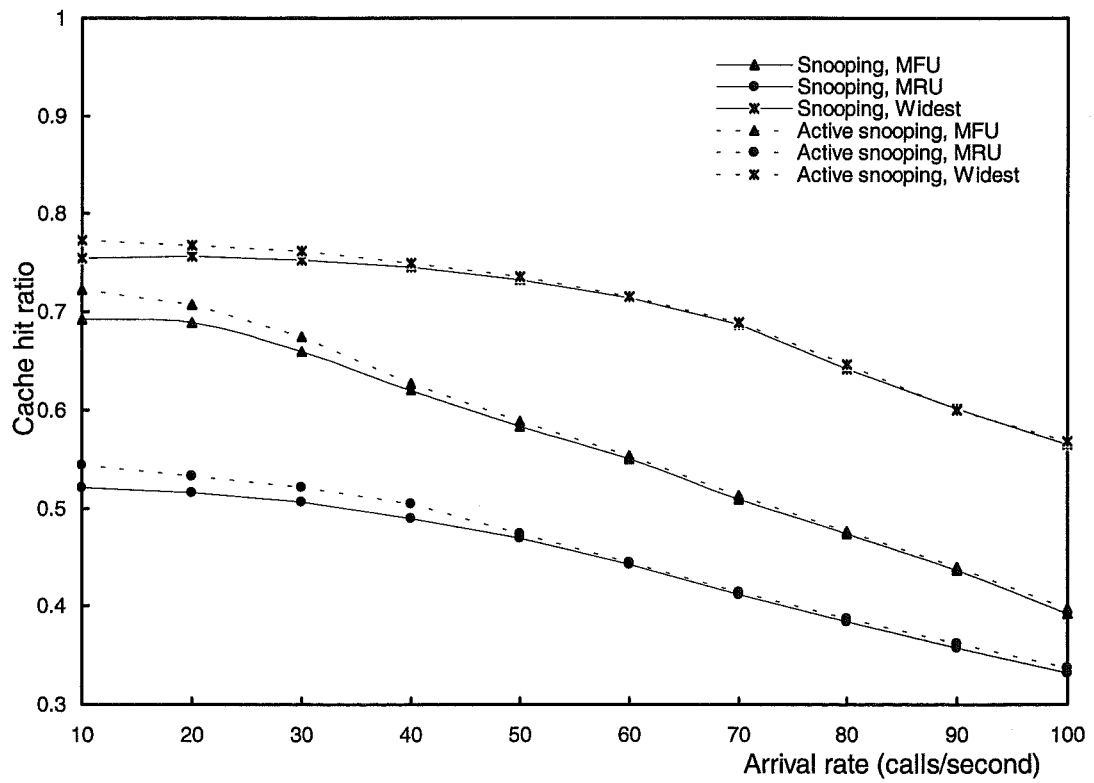


Figure 5.8-a: Impact of active snooping on the cache hit ratio, 80% voice
Figure 5.8-b: Impact of active snooping on the cache hit ratio, 80% video

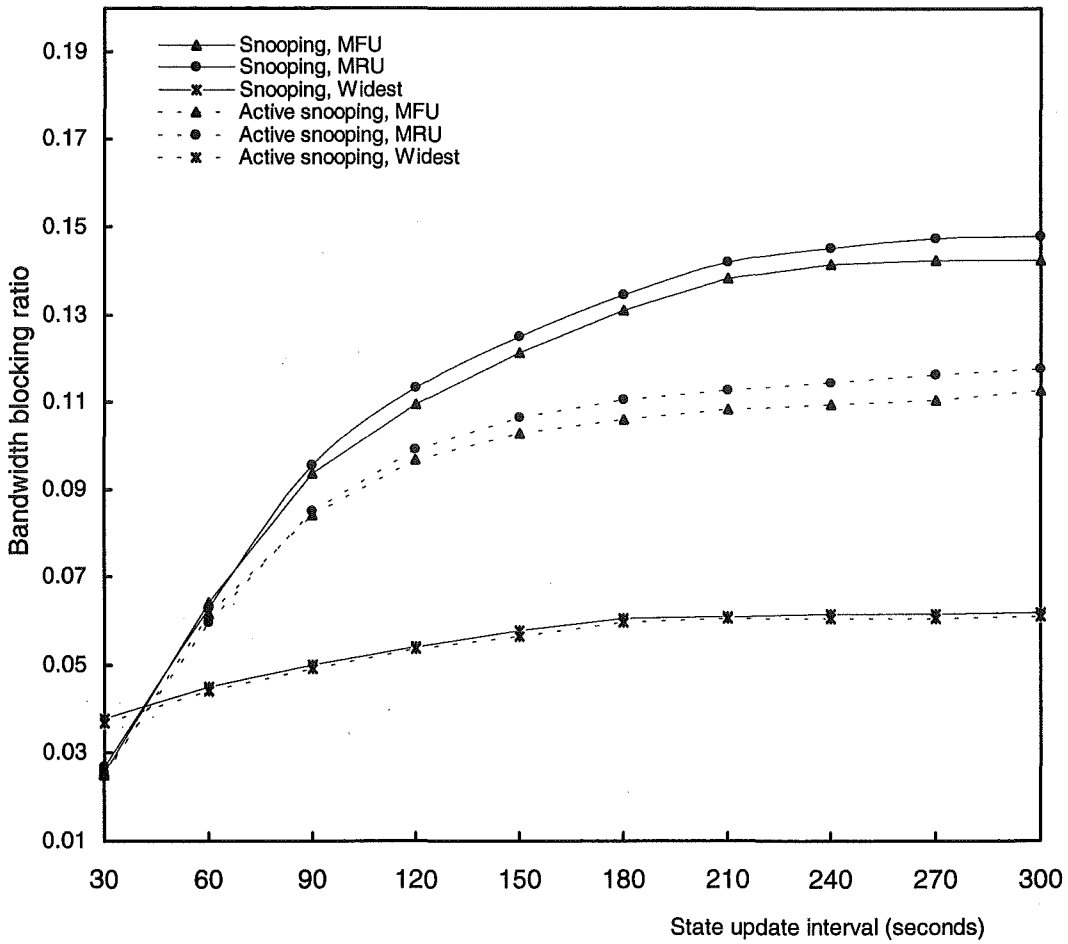


Figure 5.9: Bandwidth blocking ratio under the active snooping
Snooping interval = 30 seconds, Snooping size = 6
MRC route selection policy, LRC flushing policy, flushing size = 2

5.6.2 Performance under Inaccurate Network State Information

Active cache snooping uses segment re-computing to repair partially damaged routes. Because it is performed locally inside the network domains, segment re-computing does not suffer from the inaccuracy of the network state information caused by topology aggregation. In addition, because segment re-computing is performed by the caches at the border nodes for their local egress segments, it does not suffer from the inaccuracy of the network state information caused by the propagation delays of the network links. Therefore, active snooping is likely to reduce the bandwidth blocking ratio by reducing the likelihood of making wrong decisions. Figure 5.9 confirms that active snooping is effective in reducing the bandwidth. blocking ratio. However, we observe that under the widest snooping policy, the active

snooping does not show any advantage. Under this policy, the widest segments are selected for snooping and because of a lack of bandwidth, re-computing of wide segments is more likely to fail.

5.7 Related Work

As outlined in Chapter 1, in recent proposals on route caching in QoS capable networks, the cache architectures have been tightly coupled with the network state information maintained by the network nodes [Pay97-1, Apo98-1]. Therefore, all cache operations, including updating the cached routes, suffer from the inaccuracy of the network state information.

In contrast, our proposed distributed cache architecture has been designed in such way that it functions independently. In other words, we have de-coupled the distributed cache architecture from the network state information maintained by the border nodes. This has become feasible for two reasons. Firstly, end-to-end routes are stored in a distributed fashion in the form of several interconnected segments. Secondly, the cache element at each border node stores detailed topological information for segments. Such an approach allows cache snooping to test the cached segments by directly monitoring the corresponding links and switches inside the network domains.

Because the distributed cache architecture has been de-coupled from the network state information maintained by the border nodes, it increases the tolerance of the routing in the presence of inaccurate network state information. In other words, assuming cache snooping is in place, the accuracy of the network state information becomes less influential. This means that the network state information can be redistributed less frequently so that the overhead traffic is reduced.

5.8 Summary

In this chapter, we investigated the effectiveness of cache snooping in reducing the on-demand route computing load. In addition, we studied the impact of different snooping parameters on the performance of the distributed cache architecture. The studies conducted in this chapter have clarified several important issues that are briefly summarized as follows:

1. Cache snooping effectively contributes to reducing the on-demand route computing load. Tables 5.3-a to 5.3-c present a simplified set of results achieved under voice-dominated and video-dominated traffic. For example, one observes that for a network with nineteen domains (i.e., complete MCI backbone), under

voice dominated traffic, the core distributed cache architecture reduces the route computing load to between 18% and 9.8%. Under the same conditions, cache snooping reduces the route computing load to between 36.8% and 19.1%.

2. Cache snooping becomes less effective as the network states become more variable. The network states become highly variable at higher call arrival rates, especially under video-dominated traffic, when calls request large chunks of bandwidth. For example, as shown in Table 5.3-c, under the video-dominated traffic, assuming an arrival rate of 100 calls/second, the distributed core architecture reduces the route computing load by about 6%. By incorporating cache snooping into the core architecture, the route computing load in reduced by only 10%. *We highlight this as a weakness of cache snooping.* In Chapter 6, we show that route freezing can play a complementary role for cache snooping to alleviate this weakness.

3. Different snooping policies behave differently depending on the traffic mixture. Under voice-dominated traffic, the MFU snooping policy achieves the highest cache hit ratio, while under video-dominated traffic, the widest snooping policy works best.

4. A reasonably short snooping interval is desirable for several reasons. Firstly, it maximizes the cache hit ratio, which is translated into a higher reduction in the route computing load. Secondly, those snooping policies that achieve a higher cache hit ratio are more sensitive to the length of the snooping interval. Thirdly, the network traffic mixture can greatly affect the performance of cache snooping, especially at higher arrival rates when network states become highly variable. A shorter snooping interval can more efficiently capture and keep track of these variations. We suggest a snooping interval of 30 seconds.

5. Studies suggest that neither a very small nor a very large snooping size is adequate. After extensive investigation, we suggest a snooping size of 6 for the distributed cache architecture.

6. Assuming the snooping interval and the snooping policy are selected properly, the cache snooping increases the tolerance of the routing in the presence of inaccurate network state information caused by long network state update intervals. This is reflected by a lower bandwidth blocking ratio achieved by cache snooping.

7. Active snooping shows only a trivial advantage over normal cache snooping but it may achieve a lower bandwidth blocking ratio.

Based on the studies conducted on cache snooping, our overall conclusion is that cache snooping is quite effective in reducing the route computing load by increasing the accuracy of the cached routes. However, the snooping parameters and policies should be selected carefully so that the distributed cache architecture enjoys an increased performance with a minimal side effect on the bandwidth blocking ratio.

The studies also reveal that cache snooping is less effective in reducing the route computing load when the network state is highly variable. For example, this can happen at a high arrival rate under video-dominated traffic. In the next chapter, we show that route freezing can alleviate this problem.

Configuration of Distributed Cache Architecture	***10 calls/sec***	***50 calls/sec***	***100 call/sec***
Core Architecture + Snooping	36.8%	27.15%	19.1%
Core Architecture	18%	12.5%	9.8%

(a): Voice-dominated Traffic, Network with 19 domains

Configuration of Distributed Cache Architecture	***10 calls/sec***	***50 calls/sec***	***100 call/sec***
Core Architecture + Snooping	52.4%	38.15%	25.12%
Core Architecture	26.6%	20.1%	13.8%

(b): Voice-dominated Traffic, Network with 10 domains

Configuration of Distributed Cache Architecture	***10 calls/sec***	***50 calls/sec***	***100 call/sec***
Core Architecture + Snooping	34%	25%	10.5%
Core Architecture	18.1%	12%	6.3%

(c): Video-dominated Traffic, Network with 10 domains

Table 5.3: The impact of the cache snooping on the route computing load

Chapter 6

Performance Evaluation of Route Freezing

6.1 Introduction

The studies conducted on the effectiveness of cache snooping in reducing the route computing load in Chapter 5, indicate that under all circumstances, cache snooping loses its effectiveness at a higher network load, especially under video-dominated traffic. This is because very rapid changes in the network states can not be captured by a reasonable snooping interval so that the cache hit ratio drops at a higher network load. In this chapter, we study how route freezing can help reduce the route computing load. In addition, we show that route freezing can stabilize the cache hit ratio as the network load grows.

As detailed in Chapter 2, under the proposed distributed cache architecture, when a call ends, the tear down procedure releases allocated bandwidth for the call across the network links, and the corresponding cached route is labelled as *"available"* for potential use by subsequent calls. This approach does not guarantee that the cached route can satisfy its associated bottleneck bandwidth once the route is selected by a subsequent call, because in the time frame between a tear down procedure until the next setup procedure, changes in the network states can cause cached routes to become obsolete. Selecting obsolete routes from the cache adversely affects the routing performance. Cache snooping can improve the accuracy of the cached routes only in a statistical fashion so that the overall cache hit ratio is increased. In other words, it does not provide 100% accuracy for the cached routes. As detailed in Chapter 2, the route freezing addresses this problem by changing the normal route set up and tear down procedures so that a cached route can remain frozen for a certain period (freezing period). During the freezing period, changes in the network states do not affect the state of the route. A route may also be reused several times. In this

chapter, we study different freezing parameters including the freezing size, the freezing period, and the freezing bandwidth threshold. We investigate the trade-offs between these parameters and the performance of the distributed cache architecture.

6.2 Route Freezing vs. Route Computing Load

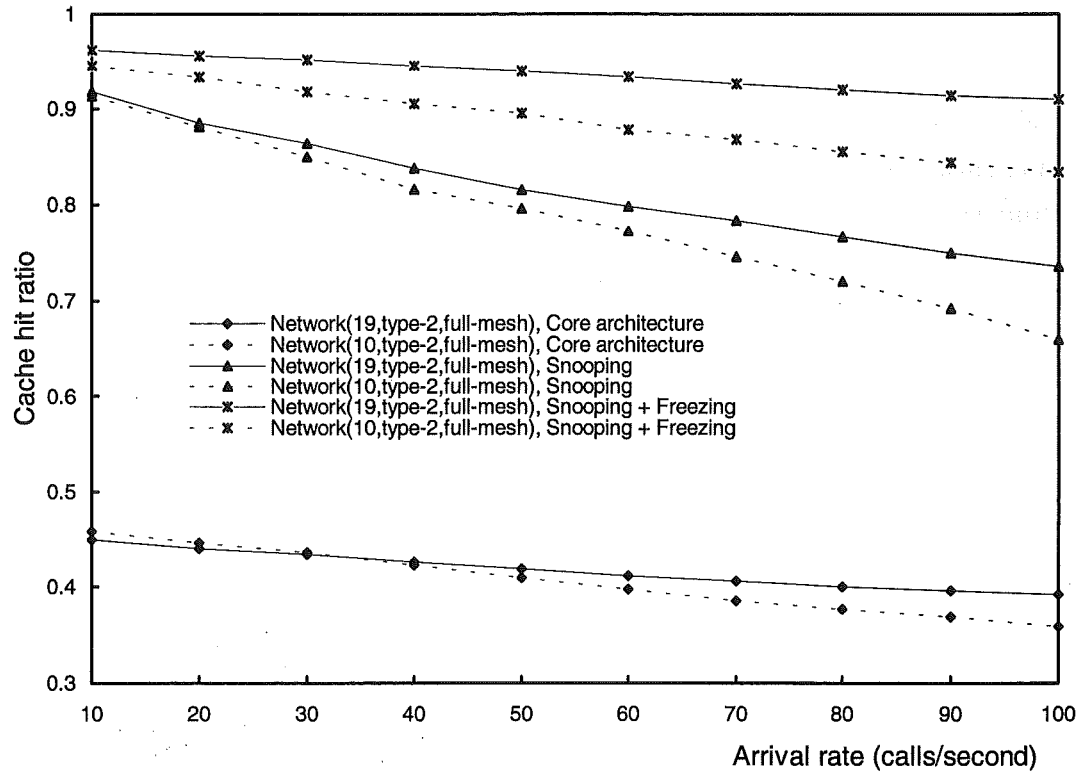
In this section, we investigate the general impact of route freezing on the cache hit ratio and the reduction achieved in the route computing load. We present our results for voice and video-dominated traffic respectively. For comparison, we present results obtained under the core architecture and cache snooping.

6.2.1 Voice-dominated Traffic

Figures 6.1 and 6.2 show the impact of the route freezing on cache hit ratio and the reduction in the route computing load under voice-dominated traffic. Note that these are the best results based on a balance between all freezing parameters. The experiments are conducted under voice-dominated traffic for two different network sizes. The main observations are:

1. As a general result, one can see that route freezing leads to a more stable cache hit ratio. In other words, under route freezing, by increasing the arrival rate, the cache hit ratio decreases more slowly when route freezing has not been used. For example, in a network with nineteen domains without snooping, by increasing the mean call arrival rate from 10 to 100 calls/second, the cache hit ratio drops by 18% (from 91% to 73%). Considering the same conditions under route freezing, the cache hit ratio drops by only 5% (from 96% to 91%). This is because the route freezing increases the cache hit ratio by providing *"bullet-proof"* frozen routes that are isolated from the changes in the network states.

2. When the network is only lightly loaded, route freezing does not make a significant difference to the cache hit ratio, because voice-dominated traffic, coupled with a low arrival rate, does not cause rapid and large changes in the network states. This implies that route freezing is useful at higher loads when network states change rapidly. In Chapter 5, we concluded that cache snooping rapidly loses its effectiveness when the network load becomes very heavy.



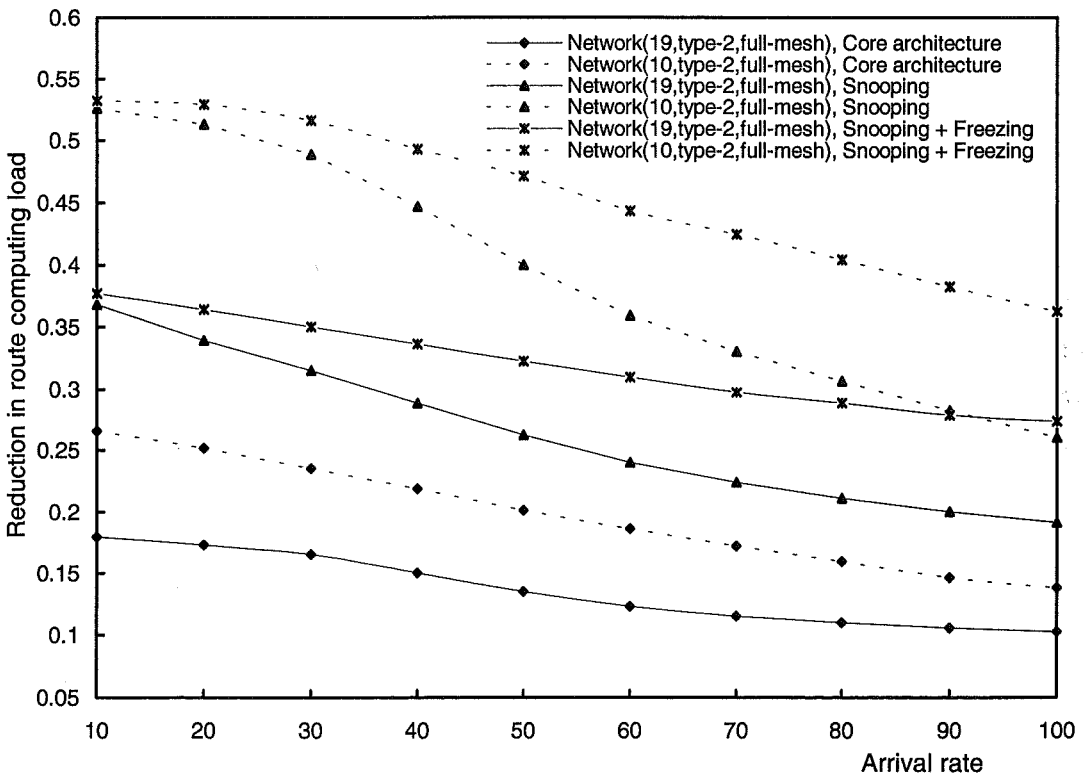


Figure 6.1: Impact of route freezing on the cache hit ratio
Figure 6.2: Impact of route freezing on the route computing load,
80% voice, Freezing size = 5, Freezing period = 400 seconds
Freezing bandwidth threshold = 2 megabits/second

3. In Figure 6.2, we observe that route freezing efficiently contributes to a reduction in the route computing load. In Table 6.1, we show a simplified set of numerical values corresponding to the graphs presented in Figure 6.2 for a network with ten domains. The values in Table 6.1 represent the amount of reduction in the route computing load. One can see that by incorporating route freezing in the distributed cache architecture, the route computing load is reduced to between 53.2% to 36.2%, depending on the arrival rate.

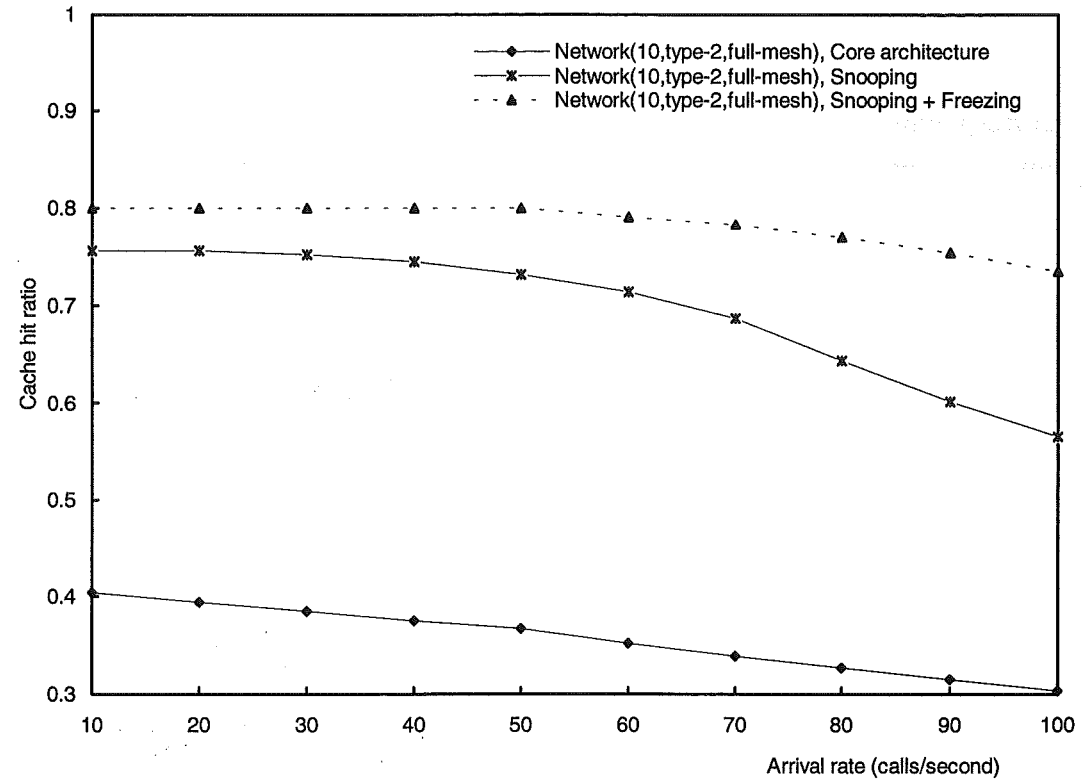
Configuration of the Distributed Cache Architecture (Network with 10 Domains)	***10 calls/sec***	***50 calls/sec***	***100 calls/sec***
Freezing + Snooping	53.2%	47.1%	36.2%
Snooping	52.6%	41%	27.2%
Core Architecture	26.5%	20%	13.8%

Table 6.1: A comparison between the reduction in the route computing load achieved by different techniques
80% voice, 20% video, Network(10, type-2, full-mesh)

6.2.2 Video-dominated Traffic

To investigate the impact of traffic conditions on the performance of route freezing, we use a video-dominated traffic mixture. Figures 6.3 6.4 present the results, assuming a network with ten domains. One observes the following:

1. Route freezing is quite effective in providing a higher cache hit ratio at higher network loads. In addition, similarly to the previous experiment with voice-dominated traffic, route freezing leads to a more stable cache hit ratio. As Figure 6.3 presents, without route freezing, by increasing the mean call arrival rate from 10 to 100 calls/second, the cache hit ratio drops by 19% (from 75.5% to 56%). If route freezing is in place, under the same conditions, the cache hit ratio drops only by 6% (80% to 74%). This reflects the effectiveness of route freezing when network states are highly variable.



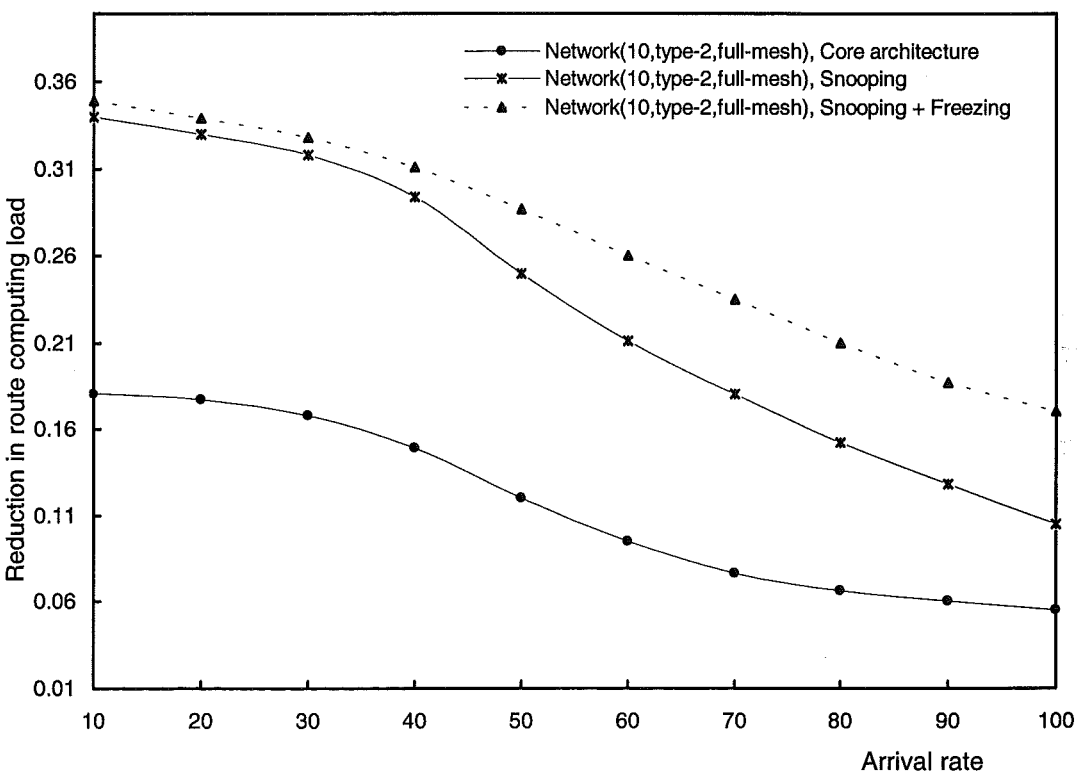


Figure 6.3: Impact of route freezing on the cache hit ratio
Figure 6.4: Impact of route freezing on the route computing load,
80% video, Freezing size = 5, Freezing period = 400 seconds
Freezing bandwidth threshold = 2 megabits/second

2. Figure 6.4 shows the effectiveness of route freezing in reducing the route computing load. In addition, Table 6.2 presents a simplified set of values corresponding to the graphs in Figure 6.4. We observe that route freezing improves the reduction in the route computing load, between 35% and 17%.

By comparing Tables 6.1 and 6.2, one sees that under voice-dominated traffic, the distributed cache architecture reduces the route computing load more efficiently. There are two reasons for this phenomenon. Firstly, the distributed cache architecture achieves a higher cache hit ratio under voice-dominated traffic. This is true even when both cache snooping and route freezing have been incorporated in the distributed cache architecture. Secondly, the cache utilization ratio is higher under voice-dominated traffic because of the much higher number of cached routes. In Chapter 2, we proposed route borrowing to increase the cache utilization ratio. We will study this technique in Chapter 7.

Configuration of the Distributed Cache Architecture (Network with 10 Domains)	***10 calls/sec***	***50 calls/sec***	***100 calls/sec***
Freezing + Snooping	35%	28.7%	17%
Snooping	34%	25%	10.5%
Core Architecture	18%	12%	5%

Table 6.2: A comparison between the reduction in the route computing load achieved by different techniques
20% voice, 80% video, Network(10, type-2, full-mesh)

6.3 Impact of Freezing Period

From Chapter 2, the freezing period (T_{freeze}) defines the time interval during which a route is kept frozen in the cache elements at the border nodes. It is worth repeating that route freezing is an end-to-end operation and is performed by individual cache elements. In this section we investigate the impact of the freezing period on the cache hit ratio. In addition, we study the impact of the freezing period on the bandwidth blocking ratio. This is important because, intuitively speaking, one can expect that a very long freezing period increases the bandwidth blocking ratio by freezing the bandwidth on the network links over a long time. The goal is to identify a realistic working range for the freezing period to maximize the cache hit ratio, leading to a

greater reduction in the route computing load, while minimizing the bandwidth blocking ratio. Note that route freezing has very negligible, or no, impact on the cache utilization ratio. Therefore, a higher cache hit ratio means more reduction in the route computing load.

6.3.1 Short Call Holding Time

We investigate the impact of the freezing period assuming a reasonably short mean call holding time of three minutes. The freezing period varies between 100 and 1000 seconds. This provides a wide range of observations so that we can draw realistic conclusions about the impact of the freezing period. Figure 6.5 presents the simulation results. We experiment using both voice-dominated and video-dominated traffic. In addition, we use different call arrival rates. The main observations can be outlined as follows:

1. Under voice-dominated traffic, when the mean arrival rate is low, the cache hit ratio is reasonably insensitive to the freezing. This is because the voice-dominated traffic, coupled with a low arrival rate does not cause significant changes in the network states. In addition, with a lower arrival rate, a frozen route is less likely to be reused. Therefore, independently of the length of the freezing period, frozen routes do not significantly increase the cache hit ratio. This is evidence that route freezing is not very effective when the network is only lightly loaded.

2. One sees that by increasing the arrival rate to 50 and then to 100 calls/second, the cache hit ratio becomes more sensitive to the freezing period. This is for two reasons. Firstly, at higher arrival rates network states change faster so that the route freezing plays a more influential role. Secondly, at higher arrival rates the frozen routes are more likely to be used. We also observe that an initial increase in the freezing period affects the cache hit ratio more significantly. In fact, after increasing the freezing period to more than 400 seconds, the cache hit ratio grows very slowly. For example, assuming a mean arrival rate of 100 calls/second, by increasing the freezing period from 100 to 300 seconds, the cache hit ratio increases from 82% to 84%. At the same time, by increasing the freezing period from 300 to 1000 seconds, the cache hit ratio increases from 84% to 85.3%. This is because a frozen route is not likely to be used very frequently. Therefore, keeping a route in a frozen state for a very long time does not increase the cache hit ratio significantly.

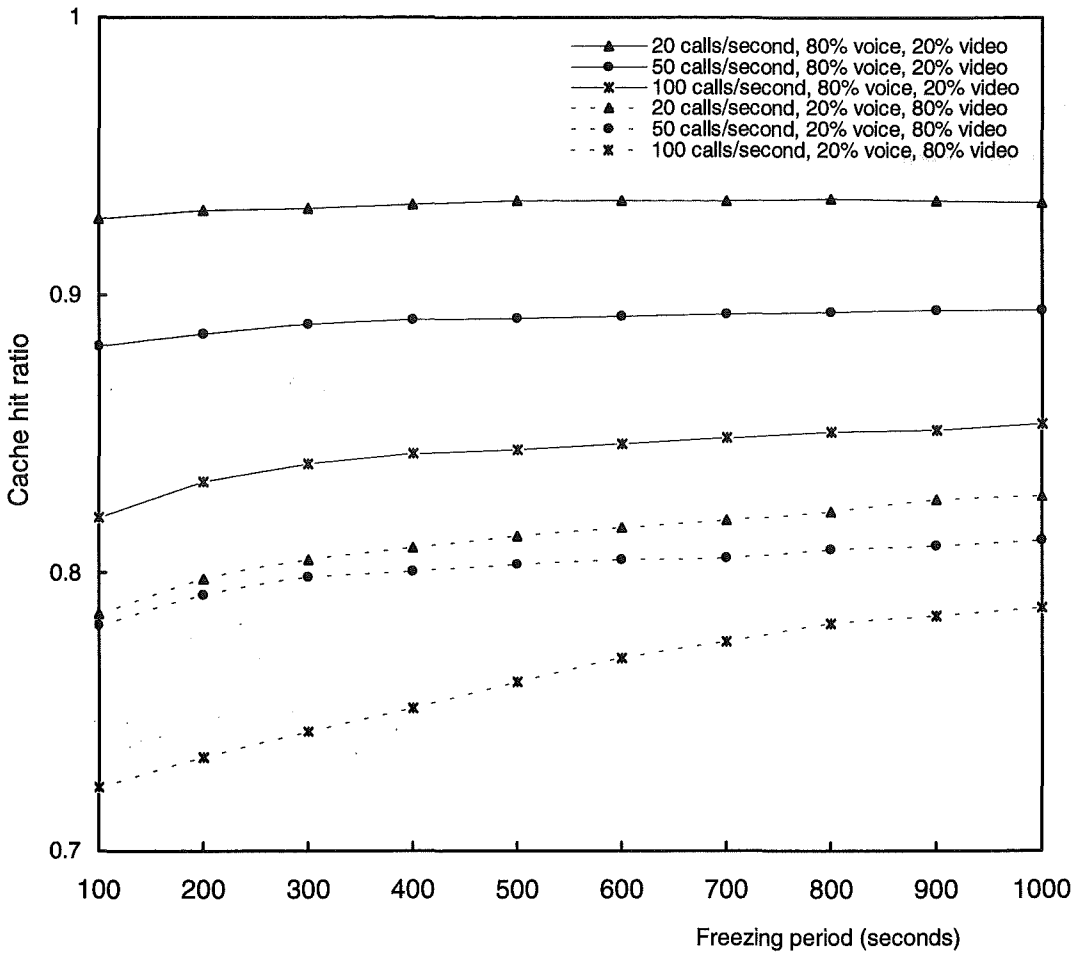


Figure 6.5: Impact of freezing period on the cache hit ratio.
Mean call holding time = 180 seconds, Freezing size = 5,
Freezing bandwidth threshold = 2 megabits/second
Network(10,type-2,full-mesh)

3. Under video-dominated traffic, the cache hit ratio is more sensitive to the freezing period, even at lower arrival rates, because there are faster and larger changes in the states. In addition, we see that at higher arrival rates, the cache hit ratio increases faster by increasing the freezing period. This is mainly because, at higher arrival rates, a frozen route is more likely to be reused. One may argue that the results indicate that under video-dominated traffic, a longer freezing period is a good choice because it leads to a higher cache hit ratio. Such a conclusion can be misleading because, as we explain in the next section, a longer freezing period, especially under video-dominated traffic, leads to a high bandwidth blocking ratio. This adversely affects the performance of the distributed cache architecture by limiting the number of cached routes. This issue has been investigated in Section 6.3.3.

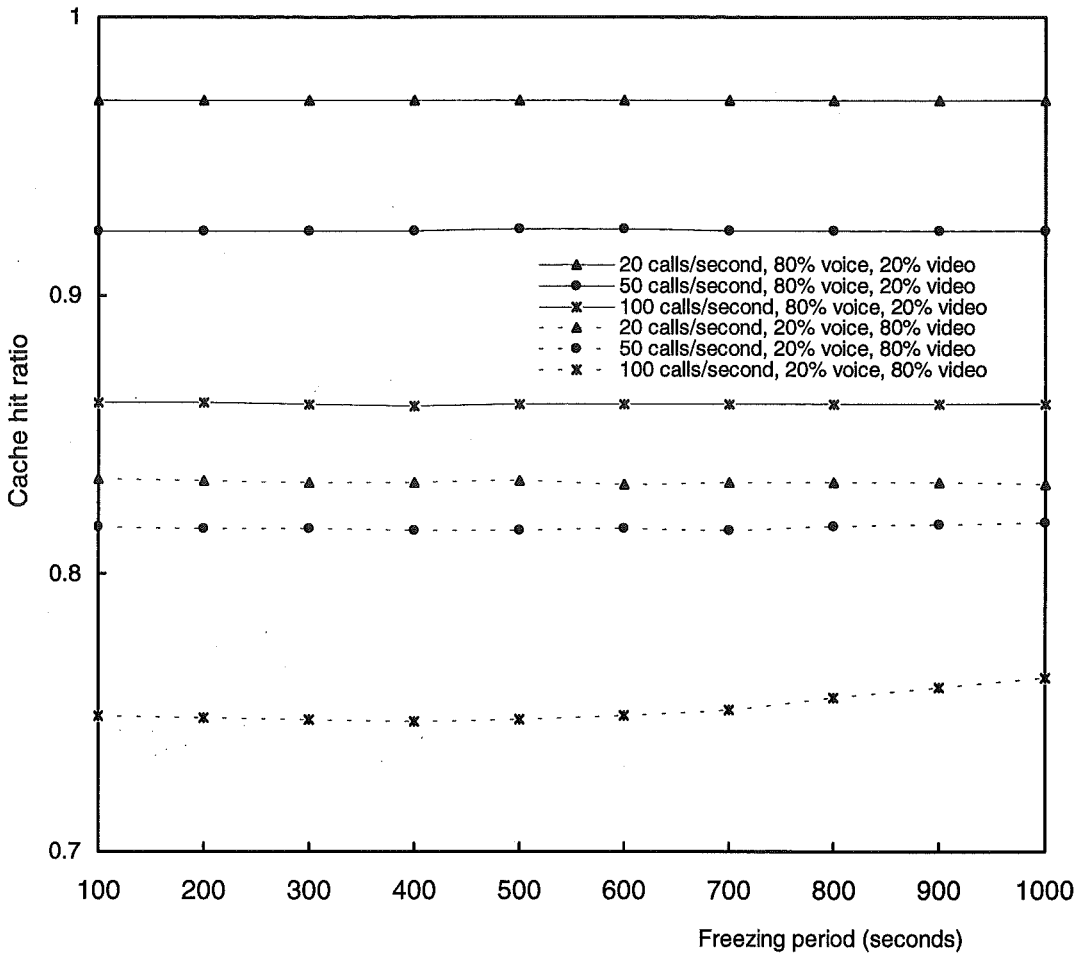


Figure 6.6: Impact of freezing period on the cache hit ratio.
Mean call holding time = 900 seconds, Freezing size = 5,
Freezing bandwidth threshold = 2 megabits/second
Network(10,type-2,full-mesh)

6.3.2 Long Call Holding Time

In Figure 6.6, we investigate the impact of the freezing period assuming a long mean call holding time of 900 seconds. Other conditions are similar to the previous experiment in Section 6.3.1. The immediate observation is that the cache hit ratio seems to be independent of the freezing period, even at higher arrival rates. There are three reasons for this phenomenon. Firstly, a long call holding time reduces the network state variations, because bandwidth is allocated and released less frequently across the network links. Secondly, the long holding time means that once a frozen route is picked by a call, it will be released after the call ends because the freezing period expires sometime during the call. This is especially the case for shorter freezing periods. Thirdly, a longer holding time reduces the likelihood of reusing the

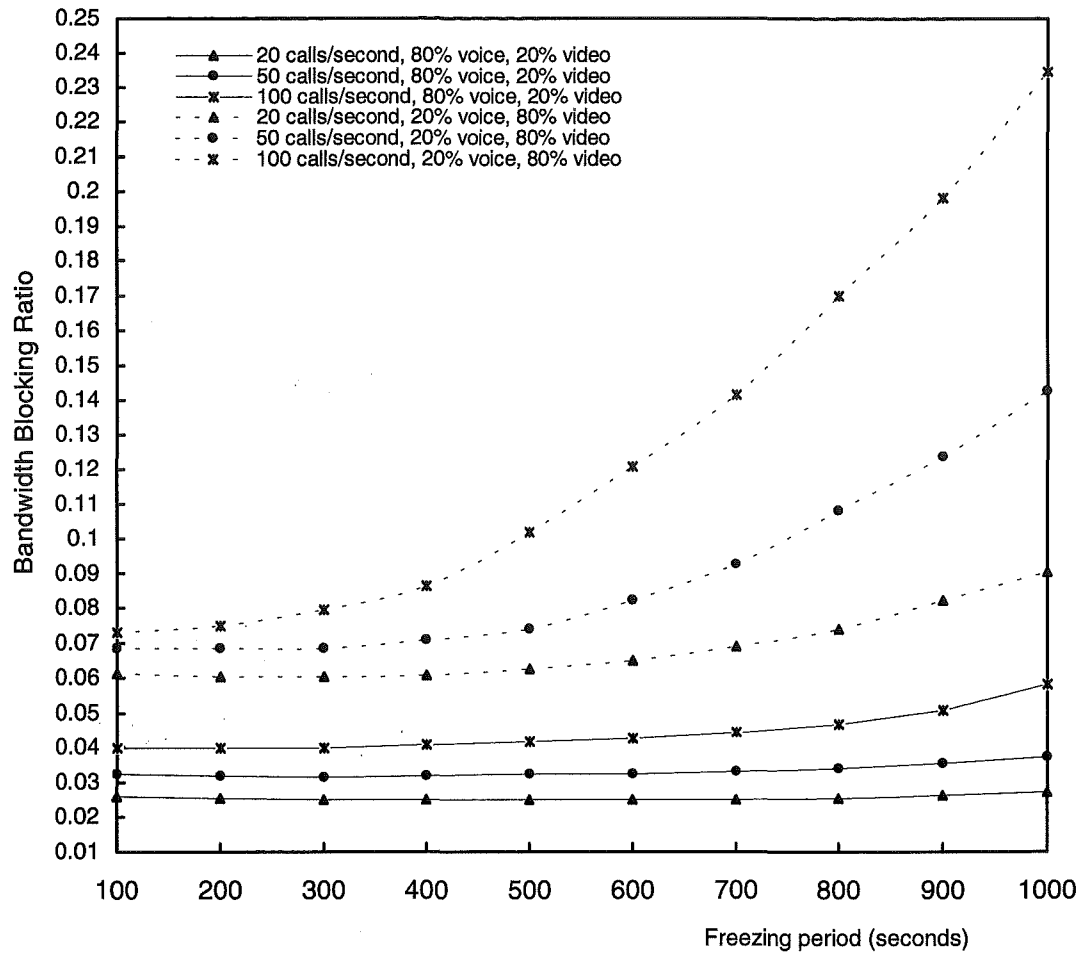


Figure 6.7: Impact of route freezing on the bandwidth blocking ratio
Freezing size= 5, Freezing bandwidth threshold= 2 megabits/second
Network(10,type-2,full-mesh)

frozen routes, because once a frozen route is selected by a call, it will be in use for a long time, limiting the chance of subsequent calls to reuse the route.

One may argue that the results in Figure 6.6.indicate that the freezing period is not important when the mean call holding time is long. Although in this case a very short freezing period is possibly harmless, choosing a long freezing period can be quite destructive because it leads to a high bandwidth blocking ratio.

6.3.3 Freezing Period vs. Bandwidth Blocking Ratio

In this section, we investigate the impact of the freezing period on the bandwidth blocking ratio. This allows us to draw realistic conclusions about the impact of the freezing period on the performance of the distributed cache architecture. Figure 6.7 presents the relationship between the bandwidth blocking ratio and the freezing

period. As can be seen, by increasing the freezing period, the bandwidth blocking ratio increases. However, one also observes that the traffic mixture plays an influential role. Under voice-dominated traffic, especially at lower arrival rates, the bandwidth blocking ratio is reasonably insensitive to the freezing period. This is because the voice routes consume little bandwidth. On the other hand, under the video-dominated traffic, the bandwidth blocking ratio increases rapidly when the freezing period is increased. For example, assuming a mean arrival rate of 100 calls/second, by increasing the freezing period from 100 to 1000 seconds, the bandwidth blocking ratio grows from 7.3% to 23.4%. This indicates that under video-dominated traffic, the freezing period should be tuned carefully.

As a general observation under all circumstances, one can see that the freezing period can be increased to 400 seconds without serious impact on the bandwidth blocking ratio.

6.4 Impact of Freezing Size

As detailed in Chapter 2, the freezing size (N_{freeze}) defines the maximum number of simultaneously frozen routes in the cache element of a border node. For example, $N_{freeze} = 2$ means that every cache element is allowed to freeze up to two routes. Note that route freezing is an end-to-end technique and is performed by individual cache elements independently. In a similar manner to the freezing period, intuitively, one expects that freezing a large number of routes in the network leads to a higher cache hit ratio. While this may be true, we need to pay attention to a potential negative impact that freezing size may have on the bandwidth blocking ratio. Therefore, in this section, we study the impact of freezing size on both the cache hit ratio and the bandwidth blocking ratio.

In Figure 6.8, we study the relationship between the cache hit ratio and the freezing size for both voice-dominated and video-dominated traffic. The freezing size varies between 2 and 20. This range is sufficiently wide to enable us to draw realistic conclusions about the freezing size. We also experiment with different arrival rates. Our observations are as follows:

1. Under voice-dominated traffic, the cache hit ratio slowly increases with an increasing freezing size. However, the initial growth of the cache hit ratio is faster. This is especially obvious at higher arrival rates. For example, assuming an arrival rate of 100 calls/second, by increasing the freezing size from 2 to 6, the cache hit ratio increases from 81.4% to 84.5%. At the same time, increasing the freezing size from 6 to 20 causes the cache hit ratio to increase from 84.5% to 87.2%. This

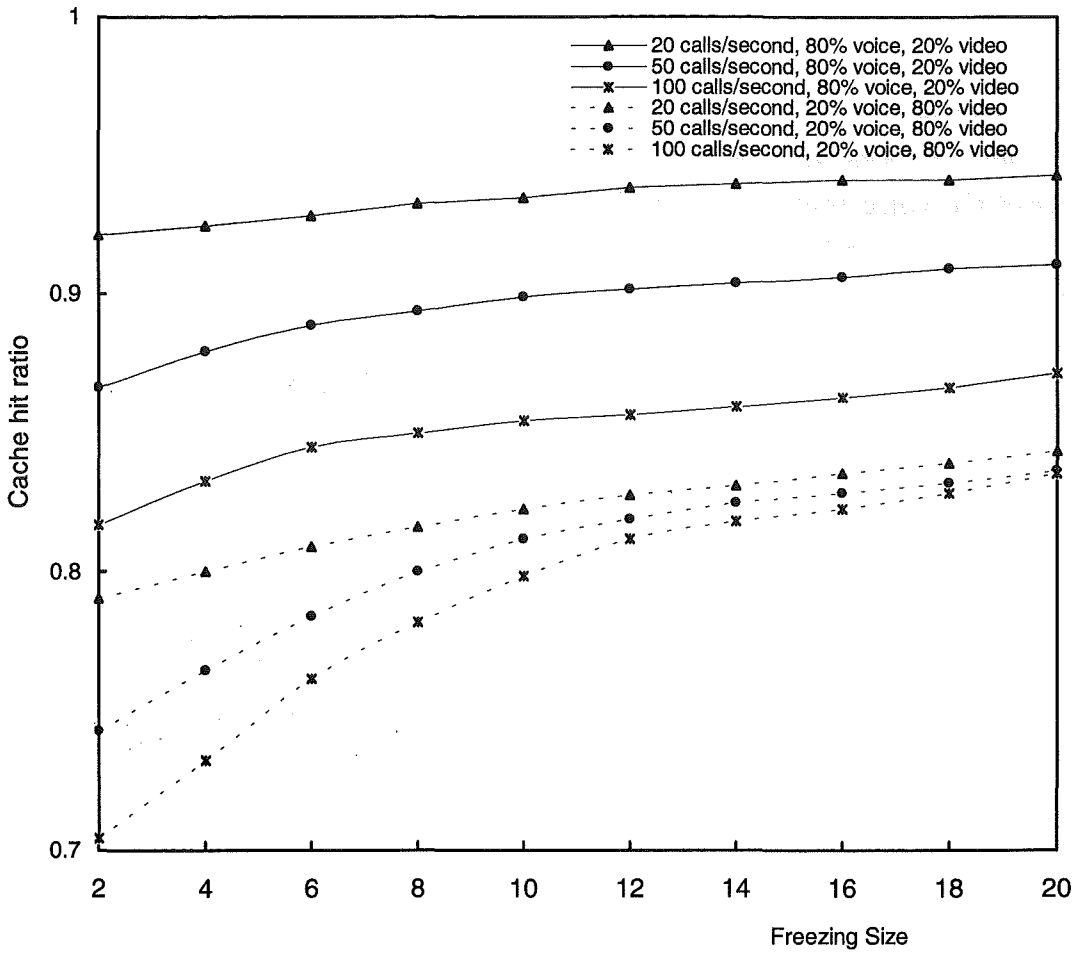


Figure 6.8: impact of freezing size on the cache hit ratio
Freezing period = 400 seconds,
Freezing bandwidth threshold = 2 megabits/second

indicates that a very large freezing size is not a good choice, because it does not lead to a much higher cache hit ratio. We can also conclude that under voice-dominated traffic, route freezing does not play an influential role because the network states do not vary significantly.

2. Under video-dominated traffic, especially at higher arrival rates, the cache hit ratio increases rapidly with increasing the freezing size, although the pace of the growth slows down at larger freezing sizes. For example, assuming an arrival rate of 100 calls/second, by increasing the freezing size from 2 to 6, the cache hit ratio increases from 70.4% to 76.2%. At the same time, increasing the freezing size from 6 to 20 causes the cache hit ratio to increase from 76.2% to 83.5%. Here, one may conclude that a very large freezing size is good because it leads to a higher

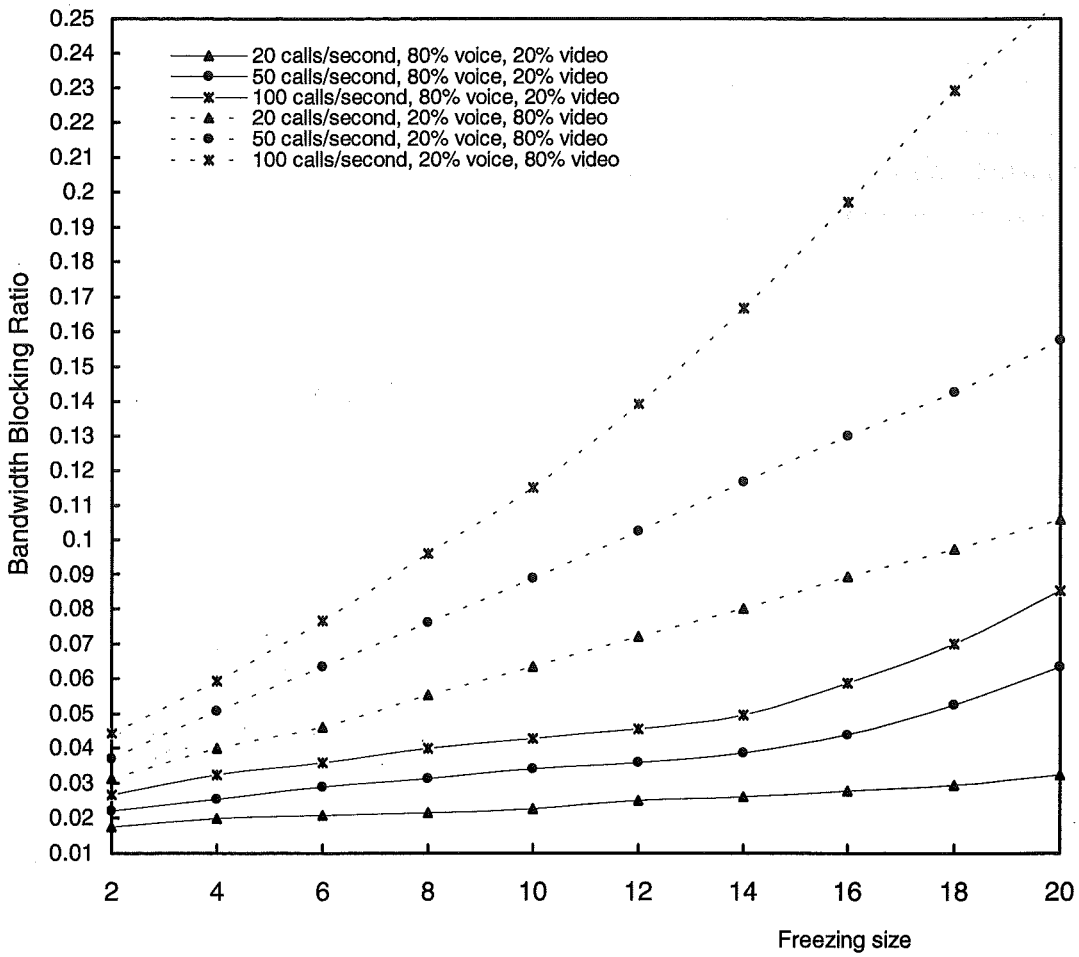


Figure 6.9: Impact of freezing size on the bandwidth blocking ratio
Freezing period = 400 seconds,
Freezing bandwidth threshold = 2 megabits/second

cache hit ratio. Similarly to the case for the freezing period, we need to consider the disadvantage of a large freezing size on the bandwidth blocking ratio.

6.4.1 Freezing Size vs. Bandwidth Blocking Ratio

In Figure 6.9, we investigate the relationship between the bandwidth blocking ratio and the freezing size. The results are somewhat similar to those obtained in Section 6.3.3, where we studied the impact of the freezing period on the bandwidth blocking ratio. One can observe that the traffic mixture, especially at higher arrival rates, significantly influences the behaviour of the bandwidth blocking ratio. Under the voice-dominated traffic, the bandwidth blocking ratio increases slowly. This is another proof that route freezing plays a vital role only when network state is highly variable. On the other hand, under video-dominated traffic, the bandwidth blocking ratio increases rapidly when the freezing size is increased. For example, assuming an

arrival rate of 100 calls/second, under video-dominated traffic, by increasing the freezing size from 2 to 20, the bandwidth blocking ratio increases from 4.4% to over 25%. This suggests that the freezing size should be carefully tuned under video-dominated traffic. As a general observation, one concludes that a freezing size of 5 can lead to a reasonably low bandwidth blocking ratio under all circumstances, while offering a good cache hit ratio.

6.5 Impact of Freezing Bandwidth Threshold

As mentioned in Chapter 2, the freezing bandwidth threshold (B_{freeze}) is a freezing parameter that selects routes for freezing; the criterion is the route bottleneck bandwidth. Therefore, each cache element selects only those routes whose bandwidth is less than the defined B_{freeze}. The goal is to investigate the relationship between this parameter and the cache hit ratio, and its effects on the bandwidth blocking ratio.

In Figure 6.10 we explore the impact of the freezing bandwidth threshold on the cache hit ratio. As a reminder from chapter 3, in our traffic model we have assumed that voice calls request a bandwidth between 16 to 64 kilobits/second, while the video calls request a bandwidth between 1 to 5 megabits/second. Following this model, the freezing bandwidth threshold varies between 1 and 5 megabits/second. Therefore, when the freezing bandwidth threshold is less than 1 megabits/second, video calls are excluded from freezing. Note that we do not exclude the voice calls from freezing because they use little bandwidth. Our main observations are as follows:

1. Under voice-dominated traffic, even at higher arrival rates, the cache hit ratio does not change significantly when the freezing bandwidth threshold is changed. This is an expected result because the freezing bandwidth threshold always includes all voice calls. However, one can still observe a slight increase in the cache hit ratio by increasing the freezing bandwidth threshold. For example, assuming an arrival rate of 100 calls/second, by increasing the freezing bandwidth threshold from 1 to 5 megabits/second, the cache hit ratio increases from 82.8% to 86%. The fact that in the voice-dominated still 20% of the arrivals are still video calls, explains this slight increase in the cache hit ratio.

2. Under video-dominated traffic, the cache hit ratio has a closer relationship with the freezing bandwidth threshold. We observe that even at lower arrival rates, the cache hit ratio increases when the freezing bandwidth threshold increases. For example, assuming an arrival rate of 20 calls/second, by increasing the freezing bandwidth threshold from 1 to 5 megabits/second, the cache hit ratio increases

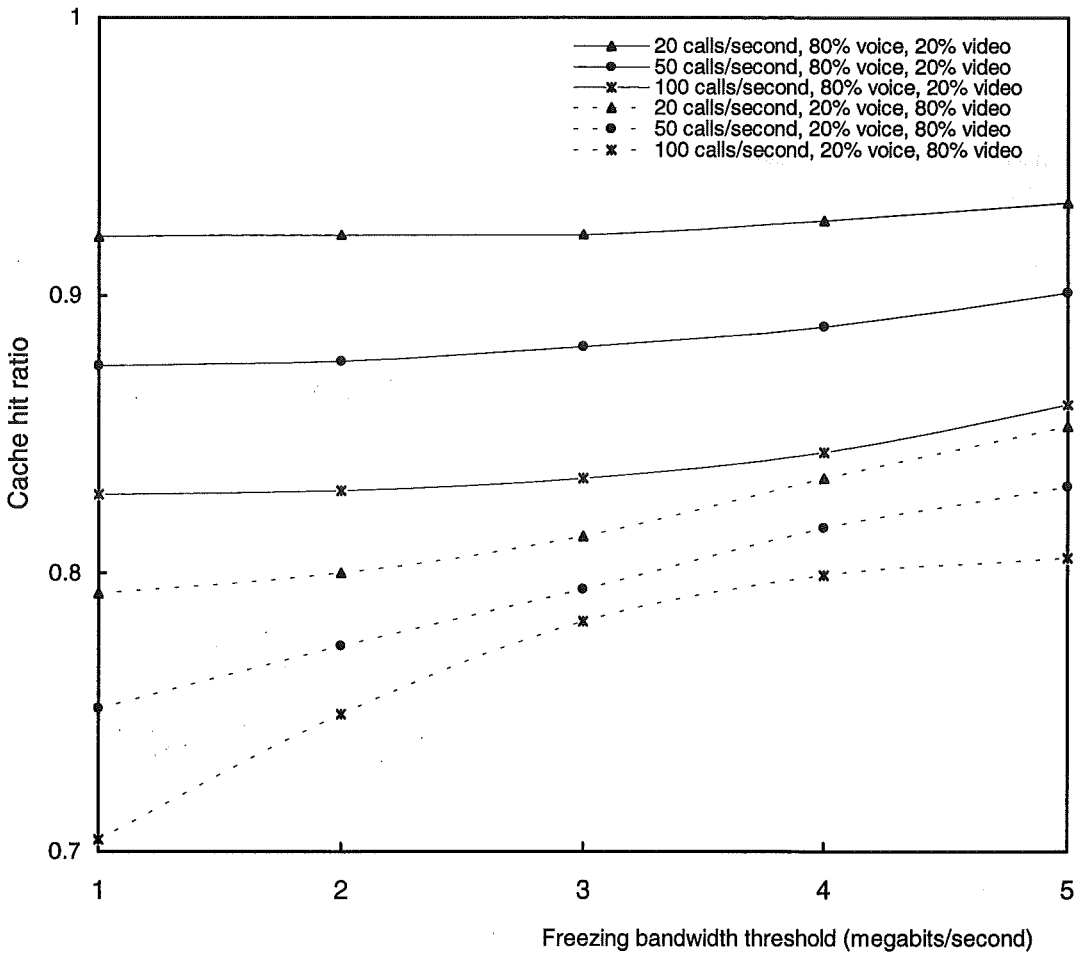


Figure 6.11: Impact of freezing bandwidth threshold on the bandwidth

Figure 6.10: Impact of freezing bandwidth threshold on the cache hit ratio
Freezing size = 5, Freezing period = 400 seconds

from 79.2% to 85.3%. This is because under a larger freezing bandwidth threshold, the cache elements select the video sessions with larger bandwidth requirements for freezing. This reduces the large variations in the network states caused by frequently allocating and releasing large chunks of bandwidth across the network links. In this way, the accuracy of the cached routes is increased and the distributed cache architecture achieves a higher cache hit ratio. Here, we do not recommend a very large freezing bandwidth threshold because it can also increase the bandwidth blocking ratio. In the next section, we study the impact of the freezing bandwidth threshold on the bandwidth blocking ratio.

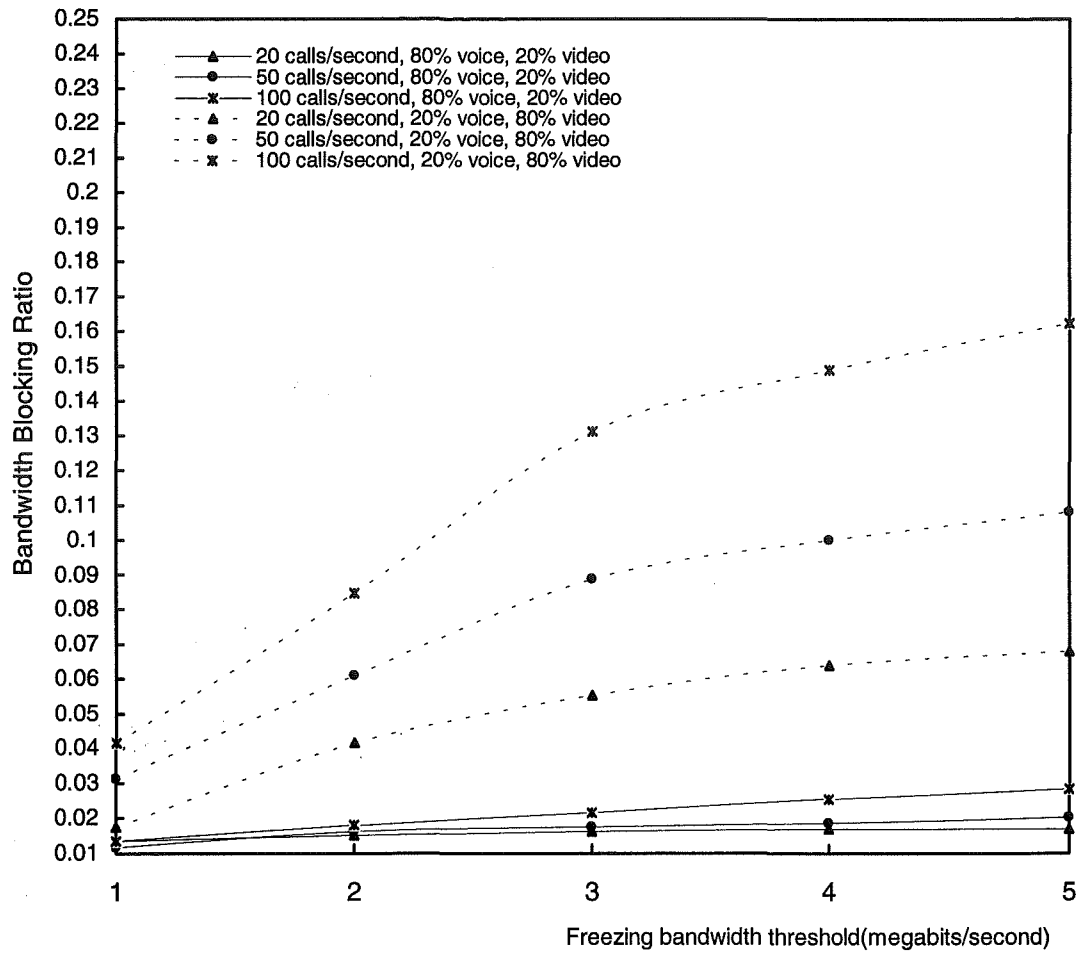


Figure 6.10:

6.5.1 Freezing Bandwidth Threshold vs. Bandwidth Blocking Ratio

In Figure 6.11, we explore the relationship between the bandwidth blocking ratio and the freezing bandwidth threshold. This is crucial to formulate realistic conclusions on the role of the freezing bandwidth threshold.

Under voice-dominated traffic, even at a higher arrival rate, the bandwidth blocking ratio shows little or no dependence on the freezing bandwidth threshold. This is for two reasons. Firstly, route freezing has little or no effect when network state is not highly variable. Secondly, the amount of requested bandwidth by voice calls is smaller than the smallest value for the freezing bandwidth threshold (i.e., 1 megabits/second), so that changing the freezing bandwidth threshold does not affect the bandwidth blocking ratio. However, we see that as the freezing bandwidth threshold is increased, the bandwidth blocking ratio is also slightly increased (about

1%). The reason is that under voice-dominated traffic, about 20% of the arrivals are video calls.

Under video-dominated traffic, the situation is quite different. The bandwidth blocking ratio increases rapidly by increasing the freezing bandwidth threshold. For example, assuming an arrival rate of 100 calls/second, by increasing the freezing bandwidth threshold from 1 to 5 megabits/second, the bandwidth blocking ratio grows from 4% to 16.2%. This is because a larger freezing bandwidth threshold increases the likelihood of selecting more bandwidth consuming video calls for freezing. In Section 6.5 we investigated the effects of the freezing bandwidth threshold on the cache hit ratio and observed that under video dominated traffic the cache hit ratio increases by increasing the freezing bandwidth threshold. Here, we observe the price of the increased cache hit ratio. A freezing bandwidth threshold of 2 megabits/seconds seems a reasonable value. It does not increase the bandwidth blocking ratio significantly, while it helps to increase the cache hit ratio under video-dominated traffic.

6.6 Summary

In this chapter, we explored the impact of route freezing on the performance of the distributed cache architecture. The goal of route freezing is to improve the cache hit ratio by providing frozen routes that are not affected by changes in the network states. In this way, route freezing can contribute to lowering the route computing load. The experiments conducted in this chapter follow two main goals.

1. The first goal is to demonstrate the general effectiveness of route freezing in reducing the route computing load in the distributed cache architecture. In addition, it shows that the route freezing is capable of increasing the cache hit ratio when the network state is highly variable and cache snooping loses its effectiveness. To this end, the results suggest that under both voice-dominated and video-dominated traffic, and for different network sizes, route freezing can reduce the route computing load. In addition, route freezing can act as a complement for the cache snooping at higher arrival rates or when the network state is highly variable. For example, as shown in Table 6.3, for a network with ten domains, under voice-dominated traffic, by incorporating route freezing in the distributed cache architecture, the route computing load is decreased to between 53.2% and 36.2%, depending on the arrival rate. Without freezing, assuming cache snooping is in place, the route computing load is decreased to between 52.6% and 27.2%. Table 6.3 also confirms similar results under video-dominated traffic.

Configuration of the Distributed Cache Architecture (Network with 10 Domains)	***10 calls/sec***	***50 calls/sec***	***100 calls/sec***
Freezing + Snooping	53.2%	47.1%	36.2%
Snooping	52.6%	41%	27.2%
Core distributed cache architecture	26.5%	20%	13.8%

(a): voice-dominated traffic

Configuration of the Distributed Cache Architecture (Network with 10 Domains)	***10 calls/sec***	***50 calls/sec***	***100 calls/sec***
Freezing + Snooping	35%	28.7%	17%
Snooping	34%	25%	10.5%
Core distributed cache architecture	18%	12%	5%

(b): video-dominated traffic

Table 6.3: Reduction in the route computing load achieved by route freezing

2. The second goal is to suggest realistic operational values for different freezing parameters including the freezing period, the freezing size, and the freezing bandwidth threshold. To this end, we investigated the impact of each parameter on the cache hit ratio and the bandwidth blocking ratio. The following are our findings on the different freezing parameters.

   - Freezing Period. The investigations on the impact of the freezing period clarify several facts. Firstly, under voice-dominated traffic, when the mean arrival rate is low, the cache hit ratio is reasonably insensitive to the freezing period. By increasing the call arrival rate, the cache hit ratio becomes more sensitive to the changes in the freezing period. Secondly, under video-dominated traffic, the cache hit ratio is quite sensitive to the freezing period, even at lower arrival rates. These observations indicate that when the network is lightly loaded, or the network state is not highly variable, route freezing does not offer any advantages. In contrast, when the network is heavily loaded or the network state is highly variable, route freezing is very effective. Thirdly, under voice-dominated traffic, the bandwidth blocking ratio is not very sensitive to the freezing period, while under video-dominated traffic, the bandwidth blocking ratio increases rapidly by increasing the freezing period. A freezing period of about 400 seconds increases the cache hit ratio without serious impact on the bandwidth blocking ratio.

- Freezing size. Firstly, under voice-dominated traffic, the freezing size does not play an influential role because network states do not vary too much. Secondly, under video-dominated traffic, especially at higher arrival rates, the cache hit ratio increases rapidly when increasing freezing size, and we need to consider the disadvantage of a large freezing size on the bandwidth blocking ratio. Thirdly, the influence of the freezing size is strongly related to the traffic mixture, especially at higher arrival rates. Under voice-dominated traffic, the bandwidth blocking ratio increases slowly, proving that route freezing plays a vital role only when the network state is highly variable. On the other hand, under video-dominated traffic, the bandwidth blocking ratio increases rapidly when the freezing size is increased. We chose a freezing size of 5. This leads to a reasonably low bandwidth blocking ratio under all circumstances, while it offering a good cache hit ratio.

- Freezing bandwidth threshold. The studies show firstly, under voice-dominated traffic, even at higher arrival rates, the cache hit ratio does change significantly when the freezing bandwidth threshold is changed. Secondly, under video-dominated traffic, the cache hit ratio has a closer relationship with the freezing bandwidth threshold. We observe that even at the lower arrival rates, the cache hit ratio increases when the freezing bandwidth threshold is increases. Thirdly, under video-dominated traffic, the bandwidth blocking ratio increases rapidly by increasing the freezing bandwidth threshold. A freezing bandwidth threshold of 2 megabits/seconds seems a reasonable value. It does not increase the bandwidth blocking ratio significantly, but increases the cache hit ratio under video-dominated traffic.

Our investigations indicate that route freezing increases the cache hit ratio so that it reduces the route computing load. Route freezing also acts as a complementary technique for cache snooping when the network state is highly variable. However, to achieve a good performance with minimal side effects, the freezing parameters should be tuned carefully.

In Chapter 5 and this chapter we have showen that the distributed cache architecture enjoys a significant increase in the cache hit ratio by incorporating cache snooping and route freezing into the core distributed cache architecture. However, one may argue that when the network load is heavy, especially under the video-dominated traffic, the distributed cache architecture does not reduce the route computing load efficiently. Such a situation is presented in Table 6.3. For a network with ten domains, under video-dominated traffic, assuming an arrival rate of 100 calls/second, when both the cache snooping and the route freezing are incorporated in the core distributed

cache architecture, the route computing load is reduced only by 17%. The main reason for such a poor performance is the low cache utilization ratio. In other words, a low cache utilisation ratio indicates that our efforts in increasing the accuracy of the cached routes are being wasted. In Chapter 7, we show that route borrowing significantly increases the cache utilization ratio. Consequently, the increased accuracy provided by cache snooping and route freezing can be used more efficiently.

Chapter 7

Performance Evaluation of Route Borrowing

7.1 Introduction

As studied in Chapters 4 and 5, when incorporated into the core distributed cache architecture, cache snooping and route freezing significantly improve the cache hit ratio by increasing the accuracy of the cached routes. In this way, they contribute to reducing the route computing load. While these techniques improve the cache hit ratio, they have no impact on the cache utilization ratio. In this chapter, we focus on improving this. Maximizing the cache utilization ratio is desirable for two reasons.

1. From Chapter 3, in the proposed distributed cache architecture, the reduction in the route computing load is defined as the produc of the cache hit ratio and the cache utilization ratio. Therefore, assuming the same cache hit ratio, a higher cache utilization ratio leads to a greater reduction in the route computing load.

2. As shown in Chapters 5 and 6, with cache snooping and route freezing in place, the distributed cache architecture enjoys a significant increase in the accuracy of the cached routes. However, cache snooping and route freezing do not affect the cache utilization ratio. In such a situation, a higher cache utilization ratio means that arriving calls can more efficiently benefit from the increased accuracy.

Under the distributed cache architecture, the cached routes can be reused only from their terminal points. As explained in Chapter 2, although simple to implement, this approach eliminates the partial usage of cached routes. This end-to-end approach,

especially in large networks with potentially long routes, reduces the cache usage and lead to more on-demand route computing. To alleviate this problem and to increase the cache utilization ratio, we proposed a technique called *route borrowing*. Route borrowing allows a call to partially borrow an end-to-end cached route instead of computing a new one. When this occurs, the whole route is labelled as *"in-use"* so that it can not be used by another call.

In this chapter, the impact of route borrowing on the cache utilization ratio and its effectiveness in reducing the route computing load have been investigated. In all experiments, we assume that cache snooping and the route freezing have already been incorporated into the core distributed cache architecture. We explore the impact of several factors on the performance of route borrowing:

1. Inter-domain network topology. This is important because in a larger network longer routes can be computed and stored in the distributed cache architecture. Longer routes cross more border nodes and may increase the performance of the route borrowing.

2. Topology aggregation techniques. In Chapter 4, we observed that the choice of topology aggregation technique significantly affects the cache utilization ratio, but has little or no impact on the cache hit ratio. Here, we explore the relationship between the cache utilization ratio and aggregation techniques under route borrowing.

3. Traffic mixture. In Chapter 4 we showed that the traffic mixture greatly affects both the cache utilization ratio and cache hit ratio. Video-dominated traffic lowers the cache utilization ratio because fewer routes can be successfully computed and stored in the distributed cache architecture. Therefore, we investigate the effectiveness of route borrowing in increasing the cache utilization ratio under video-dominated traffic.

7.2 Impact of Inter-domain Network Topology

In this section, we investigate the impact of the inter-domain network topology. We use the complete MCI backbone with nineteen domains as well as two subsets of the MCI backbone with fifteen and ten domains respectively. Figure 7.1-a shows impact of the route borrowing on the cache utilization ratio. Figure 7.1-b illustrates impact of the route borrowing on the route computing load. For comparison, we have also included the cache utilization ratio and the route computing load when route borrowing has not been used. The cache hit ratio has not been presented because it is

not affected by the route borrowing. The main observations can be outlined as follows:

1. In general, route borrowing significantly improves the cache utilization ratio. The results clearly back the intuition behind the route borrowing and prove the adequacy of the route borrowing in improving the cache utilization ratio. For example, in a network with nineteen domains, assuming an arrival rate of 100 calls/second, by incorporating the route borrowing, the cache utilization ratio increases from 26.95% to 66.2%.

2. Route borrowing is more effective in a larger network. This is especially clear in Tables 7.1-a to 7.1-c, where simplified sets of data corresponding to the graphs in Figure 7.1-a have been presented. In these tables, the cache utilization ratio has been represented in a percentage format. For example, assuming an arrival rate of 50 calls/second, one can see that in a network with nineteen domains route borrowing increases the cache utilization ratio by 41% (from 32% to 73.2%). Under the same conditions, in a network with 10 domains, the route borrowing increases the cache utilization ratio by only 28.7% (from 49% to 77.7%). The reason for this phenomenon is the presence of longer cached routes in a larger network. The longer routes are more likely to be partially borrowed by arriving calls because they cross more border nodes.

3. As shown in Figure 7.1-a, under route borrowing, there is less difference between the cache utilization ratio achieved under the different network sizes. In other words, under route borrowing the cache utilization ratio is less sensitive to the size of the inter-domain network. For example, according to Table 7.1-a and Table 7.1-c, without route borrowing, assuming an arrival rate of 50 calls/second, by changing the network size from ten to nineteen domains, the cache utilization ratio reduces by 17% (from 49% to 32%). Under the same conditions, assuming the route borrowing is in use, by changing the network size from 10 to 19 domains, the cache utilization ratio reduces by only 4.5% (from 77.7% to 73.2%). As discussed, route borrowing is more effective in a larger network because of the presence of the longer cached routes. On the other hand, a larger network in general leads to a lower cache utilization ratio. Therefore route borrowing brings the cache utilization ratio of larger networks closer to smaller networks.

4. Figure 7.1-b shows the effectiveness of route borrowing in reducing the route computing load. One sees that the route borrowing significantly contributes in reducing the route computing load. For example, as shown in Table 7.2, in a network with nineteen domains, the distributed cache architecture coupled with the cache snooping and the route freezing, reduces the route computing load to between 37.7% and 27.3%, depending on the network load. By incorporating the route borrowing, the route computing load is reduced to between 71.2% and 66.6%, depending on the network load.

Configuration of the distributed cache architecture	***10 calls/sec***	***50 calls/sec***	***100 calls/sec***
Snooping + Freezing + Borrowing	76.2%	73.2%	66.6%
Snooping + Freezing	40%	32%	26.95%

(a): Network with 19 domains

Configuration of the distributed cache architecture	***10 calls/sec***	***50 calls/sec***	***100 calls/sec***
Snooping + Freezing + Borrowing	79.6%	76.2%	69.1%
Snooping + Freezing	49.3%	41.2%	33.2%

(b): Network with 15 domains

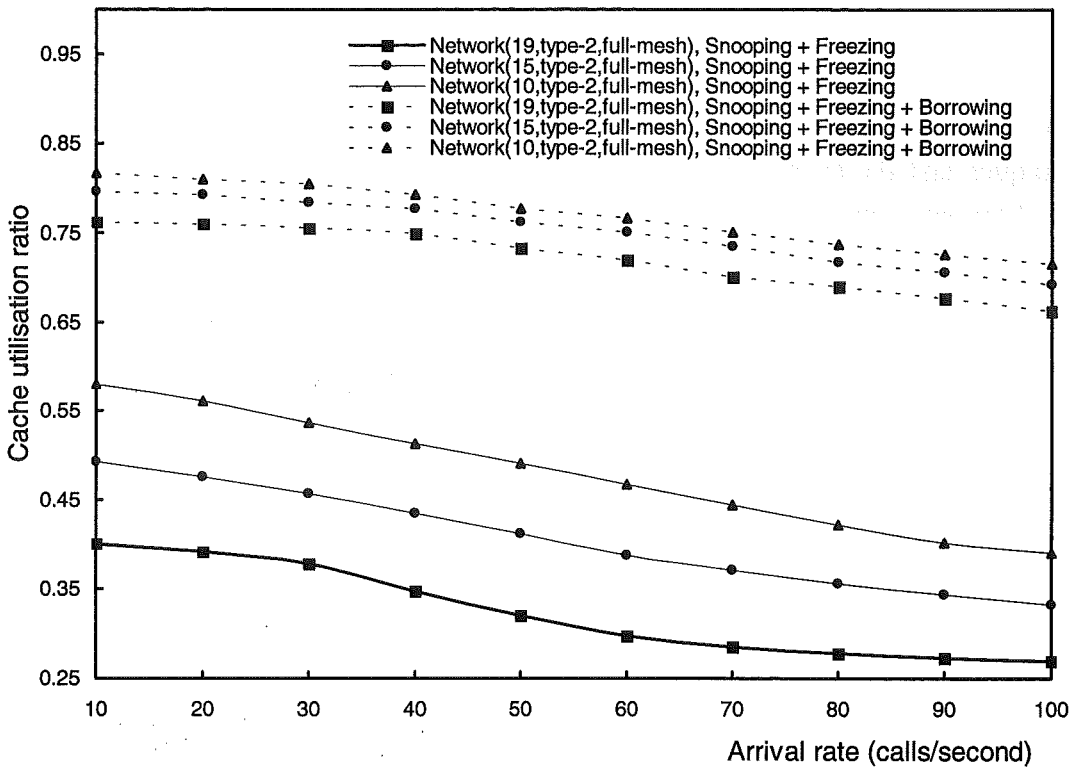
Configuration of the distributed cache architecture	***10 calls/sec***	***50 calls/sec***	***100 calls/sec***
Snooping + Freezing + Borrowing	81.7%	77.7%	71.4%
Snooping + Freezing	58%	49%	39%

(c): network with 10 domains

Table 7.1: Impact of the route borrowing on the cache utilization ratio for different network sizes (voice-dominated traffic)

Configuration of the distributed cache architecture	***10 calls/sec***	***50 calls/sec***	***100 calls/sec***
Snooping + Freezing + Borrowing	71.2%	68.7%	66.6%
Snooping + Freezing	37.7%	32.2%	27.3%

Table 7.2: Impact of the route borrowing on the reduction in route computing load (network with 19 domains)



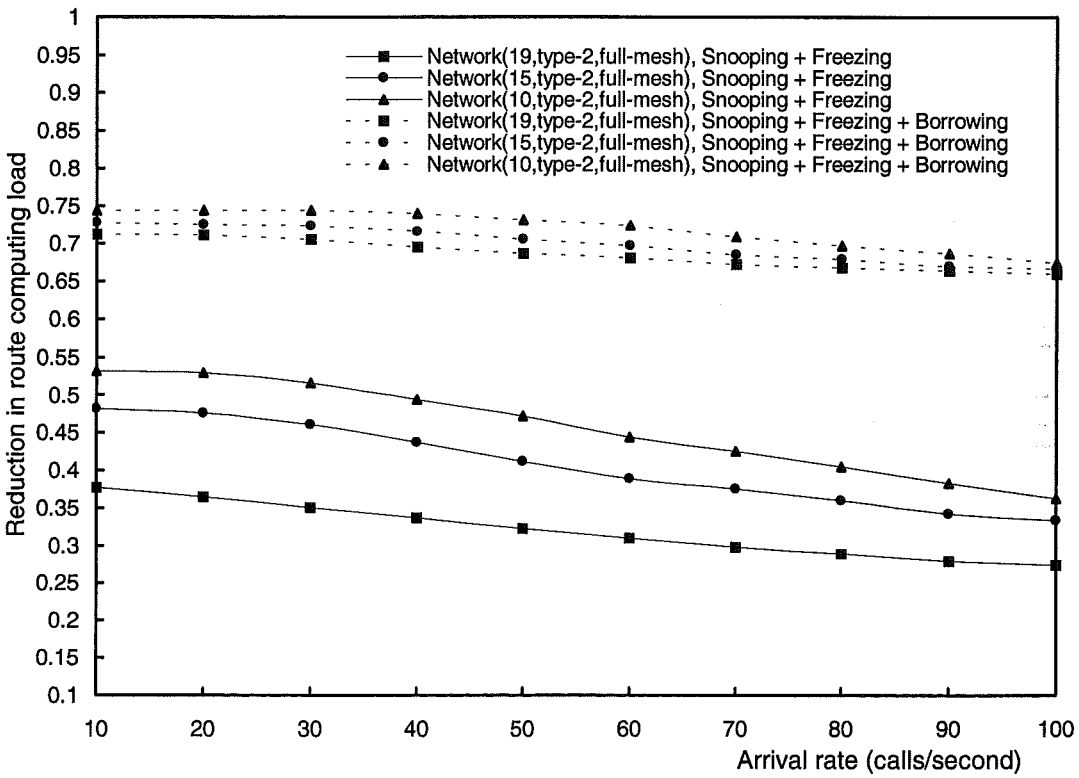


Figure 7.1-a: Impact of route borrowing on the cache utilization ratio under different network sizes

Figure 7.1-b: Impact of route borrowing on the route computing load 80% voice

7.3 Impact of Topology Aggregation

In Figure 7.2-a we explore the relationship between the topology aggregation techniques and the cache utilization ratio under the route borrowing. We have used two different network sizes with ten and nineteen domains respectively. For comparison, we also present the results without the route borrowing. Figure 7.2-b shows the influence of the topology aggregation in reducing the route computing load. The main results are as follows:

1. As Figure 7.2-a shows, independently of the adopted topology aggregation technique, route borrowing always significantly increases the cache utilization ratio. However, under route borrowing the star aggregation technique leads to a lower cache utilization ratio. This is similar to the situation where the route borrowing has not been used. As detailed in Chapter 4, the star topology aggregation reduces the accuracy of the network state and topology information more significantly than the full-mesh aggregation. Therefore the overall routing performance is reduced in such a way that fewer routes are computed and stored in the distributed cache architecture. Yet, even under the star topology aggregation, route borrowing significantly increases the cache utilization ratio. For example, in a network with nineteen domains, without route borrowing, depending on the call arrival rate, the cache utilization ratio varies between 35.4% and 20%. Under the same conditions, assuming route borrowing has been used, the cache utilization ratio varies between 71.7% and 57.7%.

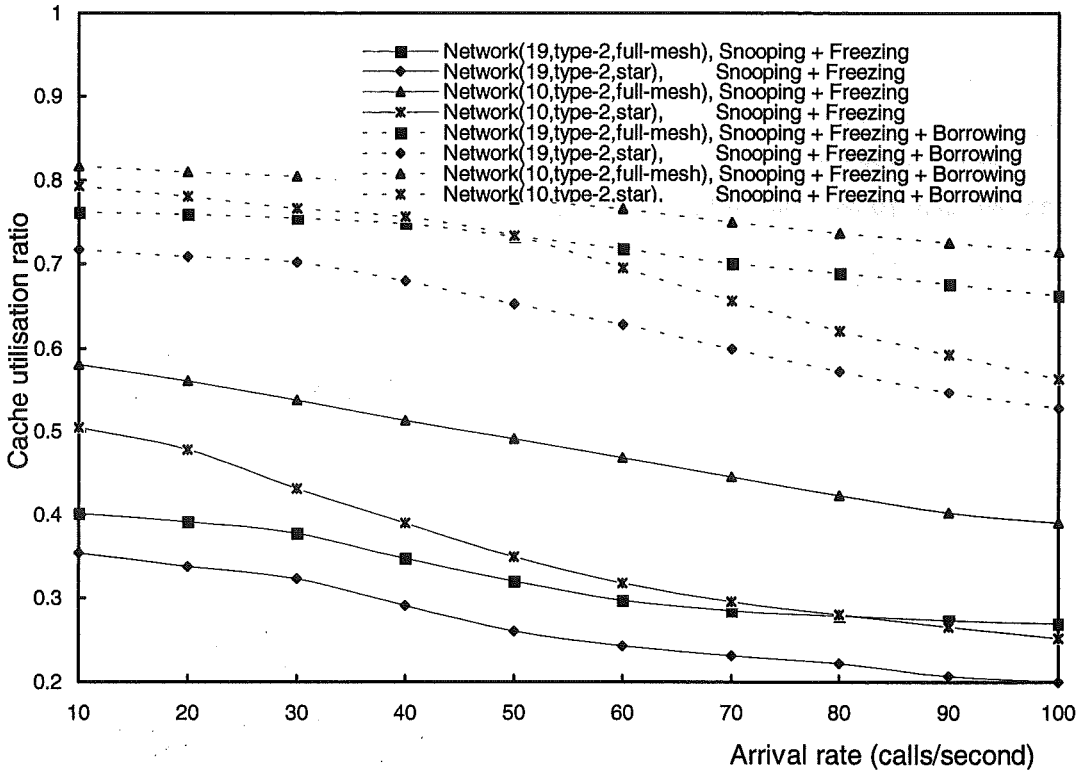
Configuration of the distributed cache architecture	***10 calls/sec***	***50 calls/sec***	***100 calls/sec***
Snooping + Freezing + Borrowing	71.2%	60.4%	52.5%
Snooping + Freezing	33.2%	23.5%	19.4%

(a): Star aggregation

Configuration of the distributed cache architecture	***10 calls/sec***	***50 calls/sec***	***100 calls/sec***
Snooping + Freezing + Borrowing	71.2%	68.4%	67.4%
Snooping + Freezing	37.7%	32.2%	27.3%

(b): Full-mesh aggregation

Table 7.3: The effectiveness of route borrowing in reducing the route computing load (Network with 19 domains)



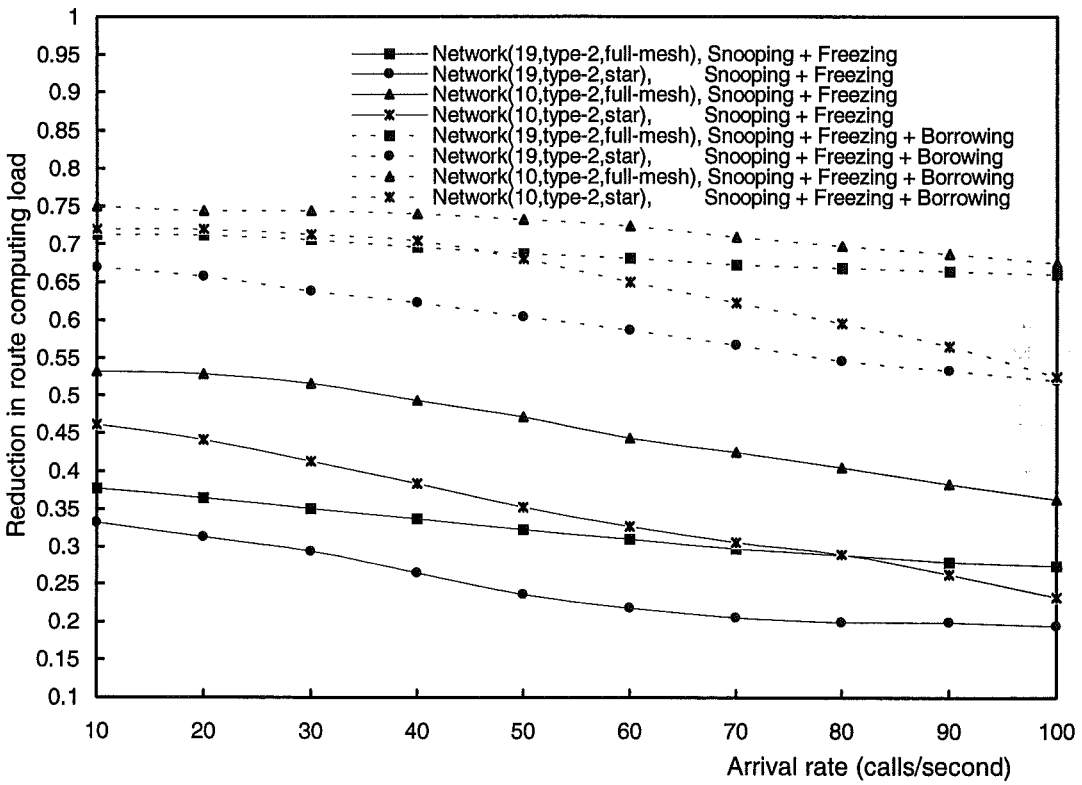


Figure 7.2-a: Impact of route borrowing on the cache utilization ratio under different topology aggregation techniques

Figure 7.2-b: Impact of route borrowing on the route computing load 80% voice

2. In Figure 7.2-b, one can see that route borrowing significantly contributes to reducing the route computing load under both star and full-mesh topology aggregation. For example, as reflected in Table 7.3-a, under the star aggregation technique, without route borrowing, the route computing load is reduced between 33.2% and 19.4%, depending on the call arrival rate. By incorporating the route borrowing, under the same conditions, the route computing load is reduced to between 71.2% and 52.5%. This clearly confirms that route borrowing is quite effective in reducing the route computing load, even in the presence of inaccurate network state information.

7.4 Impact of Traffic Mixture

As frequently discussed in previous chapters, the traffic mixture can greatly affect the performance of the distributed cache architecture in two general ways. Firstly, video-dominated traffic causes rapid and large changes in the network states so that the accuracy of the cached route is reduced. This is shown in the form of lower cache hit ratio. Cache snooping and the rout freezing have been proposed to improve the cache hit ratio in various ways. Secondly, under video-dominated traffic, fewer routes are successfully computed and stored in the distributed cache architecture. This reduces the cache utilization ratio. Here, we show that the route borrowing can improve this under video-dominated traffic.

In Figure 7.3-a, we explore the impact of route borrowing on the cache utilization ratio under different traffic mixtures. We use two networks with one and ten domains respectively. As in previous experiments, we experiment with both voice-dominated (i.e., 80% of arrivals are voice calls) and video-dominated (80% of arrivals are video calls) traffic. The main results are as follows.

1. Under video-dominated traffic, route borrowing significantly improves the cache utilization ratio. For example, as shown in Table 7.4, assuming a network with ten domains, without route borrowing and under video-dominated traffic, the cache utilization varies between 44.9% to 16.8%, depending on the call arrival rate. Under the route borrowing, the cache utilization ratio varies between 74.3% and 58.8%. One observes similar results for a network with nineteen domains.

2. Under route borrowing, the size of the network has less influence on the cache utilization ratio. Previously, in Section 7.2 we observed a similar phenomenon for voice-dominated traffic. Route borrowing is more effective in a larger network because of the presence of the longer cached routes. On the other hand, a larger network in general leads to a lower cache utilization ratio. Therefore route

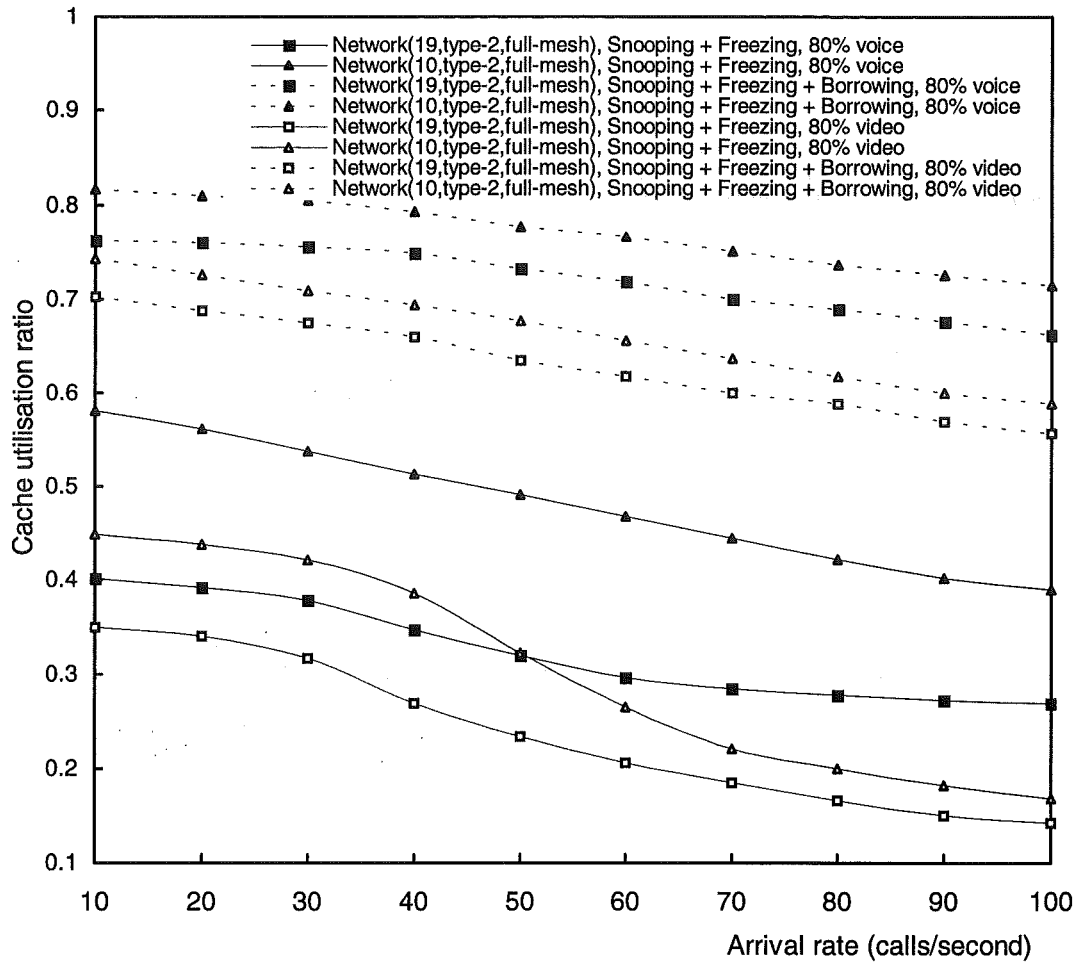


Figure 7.3-a: Impact of route borrowing on cache utilization ratio under different traffic mixtures
Network(10,type-2,full-mesh)

borrowing brings the cache utilization ratio of larger networks closer to the smaller networks.

Configuration of the distributed cache architecture (10 domains)	***10 calls/sec***	***50 calls/sec***	***100 calls/sec***
Snooping + Freezing + Borrowing	74.3%	67.7%	58.8%
Snooping + Freezing	44.9%	32.3%	16.8%

Table 7.4: The impact of the route borrowing on the cache utilization ratio under video-dominated traffic (80% of arrivals are video calls)
(Network with 10 domains)

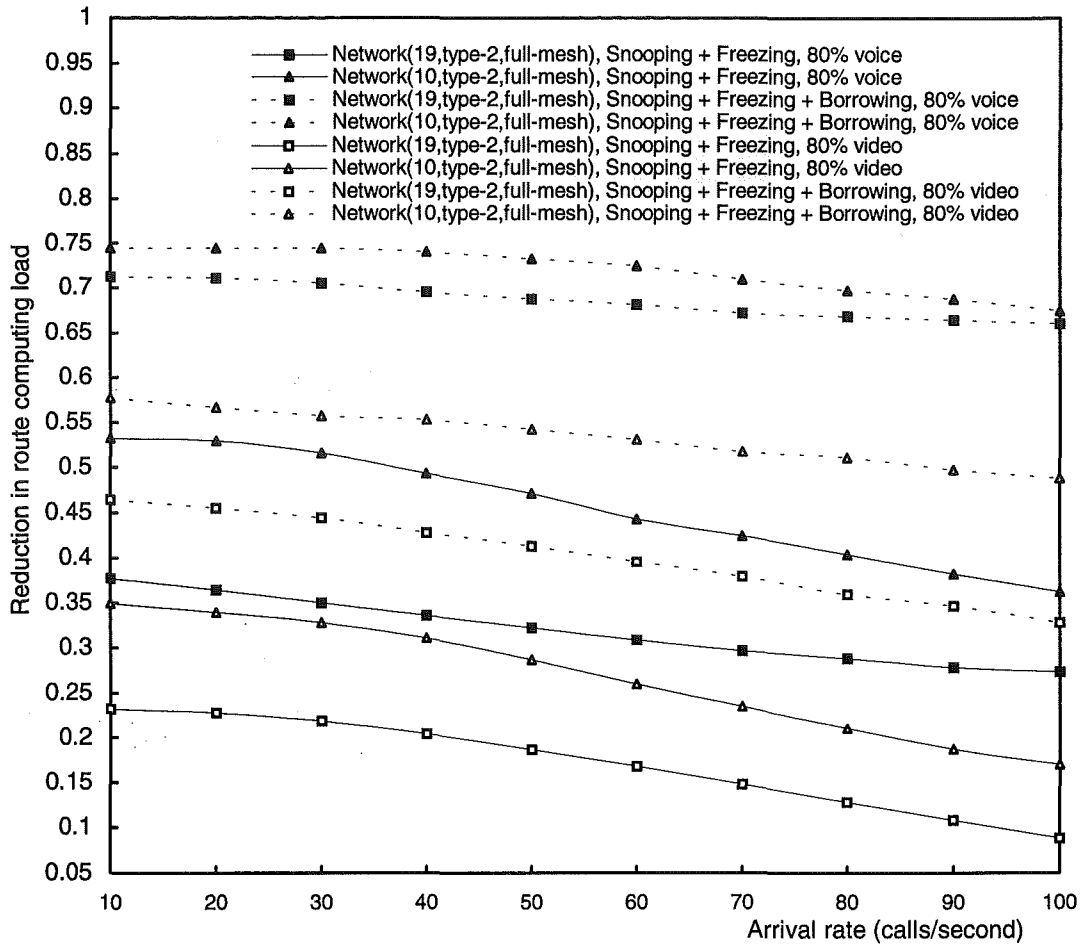


Figure 7.3-b: Impact of route borrowing on the route computing load under different traffic mixtures
Network(10,type-2,full-mesh)

In Figure 7.3-b, we explore the effectiveness of route borrowing in reducing the route computing load under different traffic mixtures. One sees that route borrowing is quite effective in reducing the route computing load under video-dominated traffic. For example, as Table 7.5 shows, for a network with ten domains, without the route borrowing, the distributed cache architecture reduces the route computing load to between 35% and 17%, depending on the arrival rate. Under route borrowing, the route computing load is reduced to between 58% and 48%.

Based on the investigations in this section, one can conclude that the route borrowing effectively contributes to reducing the route computing load under different traffic mixtures. Because of its simplicity, route borrowing can be incorporated in the distributed cache architecture with minimal effort.

Configuration of the distributed cache architecture (10 domains)	*10 calls/sec*	*50 calls/sec*	*100 calls/sec*
Snooping + Freezing + Borrowing	57.7%	54.2%	48.8%
Snooping + Freezing	34.9%	28.7%	17%

Table 7.5: The effectiveness of route borrowing in reducing the route computing load under video-dominated traffic

7.5 Summary

In this chapter, we investigated the impact of route borrowing on the performance of the distributed cache architecture. The goal is to demonstrate the effectiveness of the route borrowing in increasing the cache utilization ratio, contributing to reduce the route computing load. We explored the influence of the network topology, aggregation techniques, and traffic mixture on the performance of route borrowing. We summarize our main results as follows.

1. In general, the studies show that route borrowing significantly increases the cache utilization ratio so that the distributed cache architecture reduces the route computing load more efficiently. In addition, the route borrowing is more effective in larger networks. The studies also reveal that under route borrowing the cache utilization ratio is less sensitive to the size of the inter-domain network. These observations indicate that under route borrowing, by increasing the network size, the distributed cache architecture offers a more scalable performance in terms of the reduction in the route computing load. As a sample, Table 7.6 shows the effectiveness of route borrowing in reducing the route computing load under voice-dominated traffic for a network with 19 domains.

Configuration of the distributed cache architecture	*10 calls/sec*	*50 calls/sec*	*100 calls/sec*
Snooping + Freezing + Borrowing	76.2%	73.2%	66.6%
Snooping + Freezing	40%	32%	26.95%

Table 7.6: The reduction of the route computing load with and without route borrowing

2. Route borrowing significantly contributes to reducing the route computing load under both star and full-mesh topology aggregation. This indicates that route

borrowing is quite effective in reducing the route computing load even in the presence of inaccurate network state information.

3. Finally, the experiments show that route borrowing significantly lowers the route computing load under both the voice-dominated and video-dominated traffic. However, under voice-dominated traffic, the route computing load is reduced more efficiently.

Chapter 8

Conclusions and Future Work

8.1 Introduction

The rapid advances in telecommunication and Internet technologies is the driving force behind the emergence of a new generation of network-oriented multimedia applications such as Internet telephony and real time video applications. The QoS requirements of these applications are often one or more metrics such as bandwidth or delay. To successfully deploy these new applications, the communication networks should be QoS capable so that network resources such as bandwidth can be reserved during the lifetime of an application. The most fundamental element of a QoS-capable network is the QoS routing. A QoS routing architecture employs complicated routing algorithms to compute routes that can satisfy a set of QoS constraints. The computing load caused by frequent execution of such algorithms, especially in large networks, is a concern. Additionally, QoS routing algorithms need up-to-date network topology and state information to compute the QoS routes. This mandates regular distribution of the state and topology information across the network. The overhead traffic caused by frequent updates and redistribution of state and topology information, especially in large networks, is a concern.

In this thesis, we proposed a distributed cache architecture to reduce the route computing load caused by the execution of the QoS routing algorithms, assuming a bandwidth-based QoS model. The distributed cache architecture has been designed to easily scale to large networks. In addition, we showed that such an architecture helps to increase the robustness of QoS routing in the presence of inaccurate network state

information caused by long network state update intervals. This means that the proposed distributed cache architecture can also reduce the overhead traffic caused by the frequent distribution of the network state information, while achieving a good performance. We proposed and evaluated realistic and practical solutions that can be deployed in real life large networks. We used stochastic discrete-event simulations to evaluate the performance of the proposed distributed cache architecture. We considered realistic network topologies, routing algorithms, traffic models, and topology aggregation techniques. The remainder of this chapter presents a high level overview of the goals and achievements of this thesis.

8.2 Summary of Approach and Achievements

As detailed in Chapter 2, the proposed architecture is designed in a modular fashion. The core distributed cache architecture is the foundation of the proposal. Other modules including cache snooping, route freezing and route borrowing; techniques to improve the performance of the core distributed cache architecture in different ways. In this section we present a thorough overview of each module. We also present a brief functional overview of each module.

8.2.1 Core Distributed Cache Architecture

The core distributed cache architecture module is the foundation of the proposed architecture. It has been designed to be easily deployable in large communication networks that have been partitioned into multiple domains. In such a network model, each domain is connected to the rest of the network via several border nodes. Under the core distributed cache architecture, each border node hosts a cache element. In this way, the core distributed cache architecture can scale to large networks. These cache elements store the computed QoS routes in a distributed fashion in the form of multiple interconnected segments. Each entry in the cache element at a border node saves all necessary information about up to two segments. The core distributed cache architecture also handles operations such as route caching, route selection from the caches and cache flushing. As detailed in Chapter 2, these operations are controlled by a set of parameters and policies.

The main goal of the core distributed cache architecture is to reduce the on-demand route computing load. In an attempt to maximize its performance of the core distributed cache architecture, different parameters and policies have been associated with operations such as route selecting or cache flushing. The results of our studies point to several important issues.

Firstly, the core distributed cache architecture is effective in reducing the route computing load. For the complete MCI backbone, the distributed cache architecture reduces the route computing load by 18% to 10%. For a smaller network the route computing load is reduced more effectively.

Secondly, the size of the cache elements affects the performance of the architecture. Our studies suggested that between 70 to 100 entries per cache element offer good performance. Thirdly, the route selection policy also influences the performance, with the MRC route selection policy being the best. Fourth, the choice of topology aggregation and routing algorithms have only a minimal impact on the performance of the core distributed cache architecture. This indicates that the distributed cache architecture can be incorporated in different networking technologies and standards that may use different routing algorithms. Finally, the cache flushing affect both the cache hit ratio and the cache utilization ratio. A small flushing size of about 2, coupled with the LRC flushing policy maximizes the performance of the core distributed cache architecture.

8.2.2 Cache Snooping

Cache snooping has been proposed to alleviate the effects of the changes in the network states so that the accuracy of the cached routes is increased. The principle of cache snooping is to check the segments of the cached routes so that if a cached route is stale (because of changes in the network states) proper action can be taken. This reduces the possibility of a stale route being selected by arriving calls, by either deleting the stale routes from the distributed cache architecture or by repairing them. In essence, cache snooping is a distributed operation that is performed by each cache element independently on its local segments.

The goal of cache snooping is to increase the cache hit ratio so that the route computing load can be reduced more effectively. Additionally, cache snooping increases the tolerance of routing to the inaccuracy of the network states caused by long update intervals. Several parameters and policies control the functionality of cache snooping. These allow us tune cache snooping for maximum performance, while minimizing the possible side effects on the bandwidth blocking ratio. The following are the results of our studies:

Firstly, cache snooping effectively contributes to reducing the on-demand route computing load. If incorporated in the core distributed cache architecture, in the complete MCI backbone, the route computing load is reduced to between 37% and 19%. The core distributed cache architecture reduces the route computing load to between 18% and 9.8%. Secondly, cache snooping becomes less effective as the network states become highly variable. Thirdly, the performance of the different

snooping policies depends on the traffic mixture. Under voice-dominated traffic, the MRC performs best, while under video-dominated traffic the widest snooping policy is the best. Forthly, a reasonably short snooping interval is desirable. Studies suggest that a snooping interval of about 30 seconds is adequate. Fifth, neither a very small nor a very large snooping size is adequate. After extensive investigation, we suggest a snooping size of 6. Sixth, cache snooping also increases the tolerance of the routing in the presence of inaccurate network state information caused by long network state update intervals. This is reflected by the lower bandwidth blocking ratio achieved by cache snooping. Finally, the active snooping is effective only in lowering the bandwidth blocking ratio.

8.2.3 Route Freezing

Route freezing has been proposed for two reasons: firstly, to further improve the cache hit ratio so that the route computing load is reduced more effectively, and secondly, to act as a complement for cache snooping at higher arrival rates or when the network state is highly variable. While cache snooping increases the accuracy of the cached route in a statistical fashion, route freezing provides semi-permanent frozen routes that are not affected by the changes in the network states. Such routes are 100% accurate. Several parameters control the functionality of route freezing. Because route freezing is an inherently resource consuming operation, assigning a proper value for the freezing parameters is of paramount importance. The studies reveal the following facts about the effectiveness of route freezing and the vital role of the freezing parameters.

First, route freezing can contribute to reducing the route computing load. For the complete MCI backbone, incorporating route freezing causes the route computing load to be reduced to between 38% and 27%. Under the same conditions, the combination of the core distributed cache architecture and cache snooping reduces the route computing load to between 37% and 19%. Better results can be achieved in a smaller network. Secondly, route freezing offers only marginal advantages when the network is only lightly loaded or the network state is not highly variable. On the other hand, route freezing makes a significant difference when the arrival rate is high or the network state is highly variable. This proves that route freezing is a complement for the cache snooping. Thirdly, route freezing is not effective when the mean call holding time of calls is very long. Forth, a freezing period of about 400 seconds seems to be adequate because it maximizes the cache hit ratio with only a marginal increase in the bandwidth blocking ratio. Fifth, a freezing size of about 5 seems to be appropriate because it helps to maximize the cache hit ratio with a minimal impact on the bandwidth blocking ratio.

8.2.4 Route Borrowing

While cache snooping and route freezing focus on improving the cache hit ratio, route borrowing has been proposed to improve the cache utilization ratio. A high cache utilization ratio is important for two reasons. Firstly, a higher cache utilization ratio causes a greater reduction in the route computing load. Second, assuming the cache snooping and route freezing are in place, a high utilization ratio enables the distributed cache architecture to benefit from the added accuracy. Normally, a cached route can be reused only at its terminal points, where the route starts and ends. Route borrowing attempts to increase the cache utilization ratio by facilitating the partial usage of the cached routes from their non-terminal points. Our results suggest that the route borrowing significantly contributes to reducing the route computing load. For the complete MCI backbone, under route borrowing, the route computing load is reduced to between 76% and 66%. Without route borrowing, assuming the combination of the core distributed cache architecture, cache snooping, and route freezing are in place, the route computing load is reduced to between 40% and 27%. This clearly shows a significant improvement. Additionally, route borrowing seems to be quite effective under different traffic mixtures and topology aggregation techniques.

8.3 Future Work

This thesis has mainly focused on developing realistic and practical solutions based on a distributed cache architecture to reduce the route computing load caused by on-demand execution of QoS routing algorithms. Additionally, the tolerance of routing in the presence of inaccurate network state information is increased. Although it is an important first step towards making the large scale QoS capable networks a reality, some issues remain to be addressed while others need to be explored further.

The performance evaluation of the proposed distributed cache architecture was conducted by stochastic discrete-event simulation. This is because we believe that the simulation-based approach can give a detailed and broad perspective about the different aspects of the proposed distributed cache architecture. One may argue that an analytical model could be used. However, to be helpful, an analytical approach may require many simplifying assumptions. This may result in misleading conclusions. Clearly this is an open area for further research.

One important issue is the fault tolerance and the reliability of the proposed distributed cache architecture. Throughout this thesis, we have assumed that the cache elements are always in perfect conditions. In reality, a cache element may be unable to function properly for a variety of reasons such as hardware and software problems.

Such phenomena interrupt the normal operation of the distributed cache architecture. This is an issue that needs thorough investigation.

In this thesis, we considered a source-based routing model similar to that used in ATM standards. The Internet and IP-based networks have adopted hop-by-hop routing. The popularity of the Internet makes it a potential medium for QoS applications and therefore QoS routing. Therefore, the integration of our distributed cache architecture into IP-based networks is another direction for further research.

Throughout this thesis, we have considered a unicast point-to-point routing model. Multicast routing is another area that the distributed cache architecture may be able to cover. QoS multicast routing offers a variety of challenges. For example, the receivers on a multicast tree are likely to have heterogeneous QoS requirements. Is it possible to cache a multicast tree? Furthermore, can techniques such as cache snooping and the route freezing be extended to QoS multicast routing?

We proposed many parameters and policies to control the functionality of various operations such as route selection from the cache, cache flushing, cache snooping and route freezing. We conducted a broad range of studies on the impact of these parameters on the performance of the distributed cache architecture to find optimal values. For example, for cache snooping, we concluded that a snooping interval of 30 seconds offers a satisfactory balance between the cache hit ratio and the bandwidth blocking ratio. This static approach is practical and simple to implement. However, adaptive tuning of these parameters according to the changes in the network traffic and state of the cache elements may improve the overall performance of the distributed cache architecture. The technical feasibility and the adequacy of such adaptive techniques needs extensive and thorough investigation.

List of References

[Ach2000] Acharya, R, "A Distributed QoS Routing for Multimedia Traffic." *In Proceedings of GLOBCOM'2000*, November 2000.

[Ahn2000] Ahn, S., "Link State Aggregation Using a Shuffle-net in ATM PNNI Networks." *In Proceedings of GLOBCOM'2000*, November 2000.

[Ala95] Alaettinoglu, C., Shankar, A. U., "The Viewserver Hierarchy for Inter-domain Routing: Protocols and Evaluation." *IEEE JSAC*, vol.13, no. 8, pp. 1396-1410, October 1995.

[Apo98-1] Apostolopoulos, G., Guerin, R., Kamat, S., Tripathi, S., "Quality of service based routing: a performance perspective." *In Proceedings of ACM SIGCOMM'98 Conference,* September 1998.

[Apo98-2] Apostolopoulos, G., Tripathi, S., "On reducing the cost of on-demand QoS path computation." *In Proceedings of the Sixth International Conference on Network Protocols*, October 1998.

[Ash81] Ash, G. R., Cardwell, R. H., Murray, R. P., "Design and optimization of networks with dynamic routing." *The Bell System Technical Journal*, pp. 1787-1820, October 1981.

[Atm96] ATM Forum. "Private Network-Network Interface specification version 1.0. Technical report af-pnni-0055.000." *ATM forum*, March 1996.

[Awe93] Awerbuch, B., Azar, Y., Plotkin, S., "Throughput competitive on-line routing." *In proceedings of IEEE Symposium on Foundations of Computer Science*, pp. 32-40, October 1993.

[Awe98-1] Awerbuch, B., Du, Y., Shavitt, Y., "Routing through teranode networks with topology aggregation." *In Proceedings of ISCC'98*, pp. 406-412, 1998.

[Awe98-2] Awerbuch, B., Shavitt, Y., "Topology Aggregation for Directed Graphs." *In Proceedings of ISCC'98*, pp. 47-52, 1998.

[Awe98-3] Awerbuch, B., Du, Y., Shavitt, Y., "A simulation environment for aggregation based hierarchical topology aggregation." *In Proceedings of SCS/SPECTS'98*, July 1998.

[Awe98-4] Awerbuch, B., Du, Y., Khan, B., Shavitt, Y., "Routing through networks with hierarchical topology aggregation." *Journal of High Speed Networks*, vol. 1, no. 7, 1998.

[Bas97] Bass, T., "Internet Exterior Routing Protocol Development: Problems, Issues, and Misconceptions." *IEEE Network*, vol. 4, no. 11, pp. 50-56, July/August 1997.

[Beh98] Behrens, J., Garcia-Luna-Aceves, J. J., "Hierarchical routing using link vectors." *In Proceedings of the INFOCOMM'98*, pp. 702-710, March 1998.

[Ber92] Bertsekas, D., Gallager. R., "Data Networks." *Prentice Hall*, Second Edition, 1992.

[Bra] Braden, R., Clark, D., Shenker, S., "Integrated services in the Internet architecture: an overview", *Internet RFC 1633*.

[Cai2000] Cain., B., "Fast Link State Flooding." *In Proceedings of GLOBCOM'2000,* November 2000.

[Cal97] Calvert, K., Doar, M., Zegura, E., "Modeling Internet Topology." *IEEE Communications Magazine*, June 1997.

[Cam94] Campbell, A., Coulson, G., Hutchison, D., "A Quality of Service Architecture." *ACM Computer Communication Review*, vol. 2, no. 24, pp. 6-27, April 1994.

[Che98-1] Chen, S., Nahrstedt, K., "On finding multi-constrained paths." *In Proceedings of ICC'98*, 1998.

[Che98-2] Chen, S., Nahrstedt, K., "An Overview of Quality of Service Routing for Next-generation High-Speed Networks: Problems and Solutions." *IEEE Network Magazine*, pp. 64-78, November 1995.

[Che98-3] Chen, S., Nahrstedt, K., "Distributed QoS Routing in High Speed Networks Based on Selective Probing." *Technical report. University of Illinois at Urbana-Champaign, Department of Computer Science*, 1998.

[Cid97] Cidon, I., Rom, R., Shavitt, Y., "Multi-Path Routing Combined with Resource Reservation." *In Proceedings of INFOCOMM'97,* pp. 92-100, April 1997.

[Cra97] Crawley, E., Nair, R.., Opalan, E., Sandick, H., "A Framework for QoS-based Routing in the Internet." *IETF Internet Draft*, July 1997.

[Cor90] Cormen, T. H., Leiserson, C. E., Rivest, R. L., "Introduction to Algorithms." *The MIT Press*, Cambridge, MA, 1990.

[Dee90] Deering, S., Cheriton, D., "Multicast Routing in Datagrams Internetworks and Extended LANs." *ACM Transactions on Computer Systems*, pp. 85-111, May 1990.

[Dem89] Demers, A., Keshav, S., Shenker, S., "Analysis and Simulation of a Fair Queuing Algorithm." *In Proceedings of ACM SIGCOMM'89*, pp. 3-12, 1989.

[Diff-url] Differentiated services for the Internet. http://diffserv.lcs.mit.edu.

[Erg2000] Ergun, F., Sinha, R., Zhang, L., "QoS Routing with Performance-Dependence Costs." *In Proceedings of INFOCOMM'2000.*

[Ewi99] Ewing, G. C., Pawlikowski, K., McNickle, D., "Akaroa2: Exploiting network computing by distributed stochastic simulation." *In Proceedings of the European Simulation Multi Conference, ESM'99*, Poland, Warsaw, pp. 175-181, 1999.

[Gar95] Garcia-Luna-Aceves, J. J., Behrens, J., "Distributed, scalable routing based on vectors of link states." *IEEE JSAC*, vol. 8, no. 13, pp. 388-1395, October 1995.

[Gar79] Garey, M. R., Johnson, D. J., "Computers and Intractability: A Guide to the Theory of NP-Completeness." *W. H. Freeman*, 1979.

[Gaw95] Gawlick, R., Kalmanek, C., "On line Routing for Permanent Virtual Circuits." *In Proceedings of INFOCOM'95*, 1995.

[Gol94] Golestani., S., "A Self-Clocked Fair Queuing scheme for Broadband Applications." *In Proceedings of INFOCOM'94*, pp. 636-646, June 1994.

[Gue97-1] Guerin, R., Orda, A., "QoS-based Routing in Networks with Inaccurate Information; Theory and Algorithms." *In Proceedings of INFOCOM'97*, 1997.

[Gue97-2] Guerin, R., Orda, A., Williams, D., "QoS Routing Mechanisms and OSPF Extensions." *IETF Intenet Draft*, November 1997.

[Guo98] Guo, L., Matta, I., "On state aggregation for scalable QoS routing." *In Proceedings of the ATM Workshop'98*, pp. 306-314, May 1998.

[Gup95] Gupta, S., Ross, K. W., El Zarki, M., "On Routing in ATM Networks." *In M. E. Steenstrup*, Editor, Routing in Communications Networks, pp. 49-74. Prentice Hall, 1995.

[Hao98] Hao, F., Zegura, E. W., Bhatt, S., "Performance of the PNNI protocol in large networks." *In Proceedings of ATM Workshop*, pp. 315-325, 1998.

[Hao99] Hao, F., Zegura, E. W., "A Scalability Technique in QoS Routing." *Technical Report GIT-CC-99-04*, College of Computing, Georgia Institute of Technology, 1999.

[Hao2000] Hao, F., Zegura, E. W., "On Scalable QoS Routing: Performance Evaluation of Topology Aggregation." *In Proceedings of INFOCOMM'2000.*

[Hed88] Hedrick, C., "Routing Information Protocol." *Internet RFC 1058,* June 1988.

[Iwa98] Iwata, A., Suzuki, H., Izmailov, R., Sengupta, B., "QoS aggregation algorithms in hierarchical ATM networks." *In Proceedings of ICC'98*, 1998.

[Kap2000] Kapoor, S., "Improved Multicast Routing with Delay and Delay Variation Constraints." *In Proceedings of GLOBCOMM'2000*, November 2000.

[Kle77] Kleinrock, L., Kamoun, F., "Hierarchical routing for large networks: Performance evaluation and optimization." *Computer Networks*, 1977.

[Kom93] Kompella, V. P., Pasquale, J. C., Polyzos. G. C., "Multicast Routing for Multimedia Communication." *IEEE Transactions on Networking*, June 1993.

[Kur93] Kurose, J., "Open Issues and Challenges in Providing Quality of Service Guarantees in High-Speed Networks." *ACM Computer Communication Review,* vol. 1, no. 23, pp.6-15, January 1993.

[Kun2000] Kung, Y., "Scalability of Routing Advertisement for QoS Routing in an IP Network with Guaranteed QoS." *In Proceedings of GLOBCOMM'2000*, November 2000.

[Lac98] Lachlan, L. H., Ananda Kusuma, A. A., N., "Generalized Analysis of a QoS-aware Routing Algorithm." *In Proceedings of GLOBECOMM'98*, pp. 1-6, Sydney, Australia, November 1998.

[Laz91] Lazer, A. A., Pacifici, G., "Control of Resources in Broadband Networks with QoS Guarantees." *IEEE Communication Magazine,* vol.29 ,.no. 10, pp. 66-73, October 1991.

[Le95] le Boudec, J. Y., Przygienda, T., "A Route Pre-computation Algorithm for Integrated Services Networks." *Journal of Network and Systems Management*, vol. 3, no. 4, pp. 427-449, 1995.

[Lee91] Lee, W., Kamat, P., "Integrated packet Networks with QoS Constraints." *In Proceedings of IEEE GLOBCOMM'91*, pp.8A.3.1-8A.3.5, December 1991.

[Lee95-1] Lee, W. C., "Topology Aggregation for Hierarchical Routing in ATM Networks." *ACM Computer Communication Review*, vol. 25, no. 2, pp. 82-92, April 1995.

[Lee95-2] Lee, W. C., Hluchyj, M. G., Humblet, P. A., "Routing Subject to Quality of Service Constraints in Integrated Communication Networks." *IEEE Network Magazine*, 9(4): 46-56, July/August 1995.

[Li89] Li, K., Hudak, P., "Memory coherence is shared virtual memory systems." *ACM Transactions on Computer Systems*, Vol. 7, pp. 321-359, November 1989.

[Lor98] Lorenz, D. H., Orda, A., "QoS routing in networks with uncertain parameters." *In Proceedings of INFOCOMM'98*, pp. 3-10, 1998.

[Lor99] Lorenz, D. H., Orda, A., "Optimal partitioning of QoS requirements on unicast paths and multicast trees." *In Proceedings of INFOCOMM'99.*

[Lou91] Lougheed, V., Rekhter, Y., "A Border Gateway Protocol (BGP-3)." *Internet RFC 1267*, Oct. 1991.

[Ma97-1] Ma, Q., Steenkiste, P., "Quality-of-Service Routing with Performance Guarantees." *In Proceedings of 4th International IFIP workshop on QoS*, May 1997.

[Ma97-2] Ma, Q., Steenkiste, P., "On path selection for traffic with bandwidth guarantees." *In Proceedings on IEEE International Conference on Network Protocols*, Atlanta, Georgia, October 1997.

[Mal95] Malkin, G. S., Steenstrup, M. E., "Distance-vector routing." *In M.E. Steenstrup, editor, Routing in Communications Networks*, pp. 83-98. Prentice Hall, 1995.

[Mat95] Matta, I., Shankar, A. U., "Type-of-Service Routing in Datagram Delivery Systems." *IEEE JSAG,* vol. 13, no. 8, pp. 1411-1425, October 1995.

[Mat96] Matta, I., Shanka, A. U., "Dynamic Routing of Real-Time Virtual Circuits." *In Proceedings of INCP'96,* 1996.

[Moy91] Moy, J., "OSPF Version 2." *RFC 1247*, July 1991.

[Moy95] Moy, J., "Link-staterouting." *In M.E. Steenstrup, editor, Routing in Communications Networks*, pp. 135-157. Prentice Hall, 1995.

[Nag93] Nagarajan, R., Kurose, J. F., Towsley, D., "Allocation of local QoS constraints to meet end-to-end requirements." *In Proceedings of the Workshop on the Performance Analysis of ATM Systems*, 1993.

[Nel2000] Nelakuditi, S., Zhang, Z. L., Tsang, R. P., "Adaptive Proportional Routing: A Localized QoS Routing Approach." *In Proceedings of INFOCOMM'2000.*

[Nel90] Nelson, D. J., Sayood, K., Chang, H., "An Extended Least-hop Distributed Routing Algorithm." *IEEE Transactions on Communications*, vol. 38, no. 4, pp. 520-528, April 1990.

[Noy90] Noy, A. B., Gopal, M., "Topology distribution cost vs. efficient routing in large networks." *Computer Communications Review*, vol. 20, no. 4, pp. 242-252, 1990.

[Ock77] Ock, L. L., Kamoun, F., "Hierarchical routing for large networks: Performance evaluation and optimization." *Computer Networks*, 1977.

[Ord99] Orda, A., "Routing with end–to--end QoS guarantees in broadband networks." *In IEEE/ACM Transactions on Networking*, pp. 365-374, June 1999.

[Ord2000] Orda, A., Sprintson, A., "QoS Routing: The Precomputing Perspective." *In Proceedings of INFOCOMM'2000.*

[Pey97-1] Peyravian, M., Kshemkalyani, A. D., "Network path caching: issues, algorithms and a simulation study." *Computer Communications*, vol. 20, pp. 605-614, 1997.

[Pey97-2] Peyravian, M., Onvaral, R., "Algorithms for efficient generation of link-state updates in ATM networks." *Computer Communication and ISDN Systems*, vol. 29, no. 2, pp. 237-247, January 1997.

[Plo95] Plotkin, S., "Competitive Routing of Virtual Circuits in ATM Networks." *IEEE JSAC*, vol. 13, no. 7, pp. 1128-1136, August 1995.

[Por97] Pornavalai, C., Shiratori, N., Chakraborty, G., "QoS Based Routing Algorithm in Integrated Services Packet Networks." *In Proceedings of INCP'97*, October 1997.

[Rek95] Rekhter, Y., Li, T., "A Border Gateway Protocol 4 (BGP-4)." *Internet RFC 1771*, March 1995.

[Sal97] Salama, H. F., Reeves, D. F., Viniotis, Y., "A Distributed Algorithm for Delay-Constrained Routing." *In Proceedings of INFOCOMM'97*, April 1997.

[Sha98] Shaikh, A., Rexford, J., Shin, K., "Efficient Precomputation of Quality-of-Service Routes." *In Proceedings of Workshop on Network and Operating Systems Support for Digital Audio and Video*, July 1998.

[Shi95] Shin, K. G., Chou, C. C., "A Distributed Route-Selection Scheme for Establishing Real-Time Channel." *In Proceedings of 6th IFIP Conference on High Performance Networking*, pp. 319-329, September 1995.

[Su2000] Su, X., "Source Routing in Networks with Uncertainty: Interference, Sensitivity, and Path Caching." *In Proceedings of GLOBECOMM'2000*, November 2000.

[Tan96] Tanenbaum, A. S., "Computer Networks." Prentice-Hall, Upper Saddle River, NJ, 1996.

[Tsa89] Tsai, W. T., Ramamoorthy, C. V., Tsai, W. K., Nishiguchi, O., "An adaptive hierarchical routing protocol." *IEEE Transactions on Communications*, 38(8): 1059-1075, August 1989.

[Tsu88] Tsuchiya, P. F., "The landmark hierarchy. A new hierarchy for routing in very large networks." *ACM Computer Communication Review*, vol. 18, no. 4, pp. 35-42, August 1988

[Vog96] Vogel, R., Herrtwich, R. G., Kalfa, W., Wittig, W., Wolf, L. C., "QoS-based Routing of Multimedia Streams in Computer Networks." *IEEE JSAC*, vol. 14, no. 7, pp. 1235-1244, September 1996.

[Wan95-1] Wang, Z., Crowcroft, J., "Quality-of-Service Routing for Supporting Multimedia Applications." *IEEE JSAC*, vol. 14, no. 7, pp. 1288-1234, September 1995.

[Wan95-2] Wang, Z., Crowcroft, J., "Bandwidth-delay based routing algorithms." *In Proceedings of GLOBECOMM'95*, pp. 2129-2133, November 1995.

[Wha95-1] Whay, L. C., "Spanning tree method for link state aggregation in large communication networks." *In Proceedings of INFOCOMM'95*, pp. 297-302, 1995.

[Wha95-2] Whay, L. C., "Topology aggregation for hierarchical routing in ATM networks." *In Proceedings of ACM SIGCOM'95*, pp. 82-92, 1995.

[Zha90] Zhang, L., "Virtual Clock: A New Traffic Control Algorithm for Packet switching Networks." *In Proceedings of ACM SIGCOMM'90*, pp. 19-29, September 1990.

[Zha93] Zhang, L., Deering, S., Estrin, D., Shenker, S., Zapala, D., "RSVP: A New Resource Reservation protocol." *IEEE Communications Magazine*, vol. 31, no. 9, pp. 8-18, September 1993.